Universidad Zaragoza
1542

# Trabajo Fin de Máster

## Calibración de un algoritmo de optimización multiobjetivo en problemas de secuenciación de operaciones

## Calibration of a multi-objective optimization algorithm in operation sequencing problems


Autor:

Pablo Gutiérrez Ranera

Directora:

Mª José Oliveros Colay

ESCUELA DE INGENIERÍA Y ARQUITECTURA


2018

# Calibración de un algoritmo de optimización multiobjetivo en problemas de secuenciación de operaciones

## RESUMEN

En este trabajo de final de máster se profundiza en la parametrización y análisis de los resultados obtenidos con un algoritmo de colonia de hormigas resolviendo problemas complejos de secuenciación de operaciones multicriterio.

En primer lugar se implementa la opción de resolver las secuencias en modo permutation. Luego se toma la decisión de qué métricas utilizar para realizar una correcta valoración de los resultados y se desarrolla una herramienta para poder obtener el valor de esas métricas de forma automática.

Por último se lleva a cabo la experimentación, que consiste en una búsqueda de la configuración idónea de los parámetros para la obtención de la mejor optimización multicriterio, y una análisis comparativo entre la resolución en modo permutation y modo non-permutation.

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

# DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

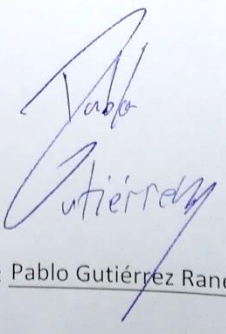
D./Dª. Pablo Gutiérrez Ranera,

con nº de DNI 77132240-S en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) Máster, (Título del Trabajo) Calibración de un algoritmo de optimización multiobjetivo en problemas de secuenciación de operaciones,

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 21 de noviembre de 2018



Fdo: Pablo Gutiérrez Ranera

# Índice

# 1. Objetivos y alcance

## 1.1 Antecedentes

Durante el trabajo para la finalización de mi Grado en Ingeniería de Tecnologías Industriales (Gutiérrez, 2016) estudié y analicé los resultados obtenidos por un algoritmo basado en colonias de hormigas para la resolución de problemas de secuenciación de operaciones o scheduling.

En dicho trabajo arreglé y mejoré el propio código, dotándolo de herramientas con las que poder hacer más simulaciones, corrigiendo fallos que lo hacían un código más robusto, así como adecuando la visualización de los datos para un análisis de estos más cómodos.

Debido al desconocimiento inicial del algoritmo ante los problemas propuestos, gran parte del trabajo estuvo dedicado a la parametrización del algoritmo (que luego se concluyó que era un paso muy necesario para la obtención de unos resultados competentes) para la resolución monobjetivo en resoluciones de problemas en modo non-permutation. Se compararon dichos resultados con los de otros autores y se concluyó que las capacidades del algoritmo en problemas pequeños y pequeños tiempos de computación eran muy grandes.

Ya que el algoritmo también estaba diseñado para la resolución de problemas multicriterio se testearon sus capacidades en bancos de problemas propios y se corroboró un correcto funcionamiento, aunque no se alcanzó la concreción necesaria en el análisis de los resultados para comparar las soluciones con diferentes configuraciones de parámetros ni con los resultados de otros autores.

En la literatura hay una gran cantidad de estudios y análisis sobre la resolución de problemas de secuenciación de operaciones, o scheduling, que se han visto reforzados por los avances en cálculo computacional durante los últimos años. Sin embargo, apenas se encuentra literatura de resoluciones de problemas mediante la optimización multicriterio en enunciados con más detalles o restricciones que los acerquen a la realidad. Así, la mayoría de estos trabajos tratan problemas simples desde el punto de vista industrial, sin apenas condicionantes propios de la industria.

Por otro lado, y con el desarrollo de los algoritmos metaheurísticos, cada vez se utilizan más éstos para resolver problemas de secuenciación, y otros de naturaleza combinatoria. La cuestión es que estos algoritmos quedan definidos por unos elementos básicos y comunes, así como por unos parámetros. En la mayoría de los casos, estos parámetros toman valores que otros autores han utilizado o tras unas reducidas pruebas previas, sin reflexionar qué valor tienen que tener dichos parámetros para mejorar la eficiencia del algoritmo.

## 1.2 Objetivos

Por tanto, el objetivo principal de este trabajo es el análisis del funcionamiento de este algoritmo en problemas multicriterio (tanto resolviéndolo en modo permutation, más estudiado en la literatura, como en modo non-permutation), adoptando, para el análisis, las métricas más competentes que analizan las resoluciones multicriterio.

Además, para una mejor comprensión del funcionamiento del algoritmo resulta necesario implementar el registro de la información con la que trabaja el código, para visualizar, en mayor medida, las capacidades de éste en la resolución de problemas monobjetivo en el tipo de problemas objeto de este trabajo.

Así, el principal objetivo de este trabajo es estudiar el comportamiento del algoritmo de la colonia de hormigas para enunciados cercanos a la realidad con optimización multicriterio, lo cual supone un análisis exhaustivo de la importancia de la calibración de los parámetros que definen el algoritmo en distintos tipos de problemas de secuenciación.

## 1.3 Alcance

Para llevar a cabo los objetivos propuestos se ha requerido la modificación del programa (en lenguaje C++) para corregir errores que impedían algunas de las simulaciones, así como para representar la información necesaria para el análisis de los resultados y, por último, añadir la capacidad de resolver los problemas en otro modo al prediseñado (en modo permutation).

Dicho análisis, por la cantidad de resultados y complejidad de valoración de éstos, ha requerido la creación de una herramienta en el entorno de Excel (mediante programación en Visual Basic) que ha permitido evaluar, siguiendo las métricas encontradas en la literatura, los conjuntos de soluciones obtenidos por el algoritmo, así como su representación, para una valoración y comparación con los resultados de otros autores tanto visual como cualitativa.

La valoración del funcionamiento del algoritmo requerirá el uso de problemas empleados en la literatura, y un desarrollo de la experimentación con diferentes baterías de configuraciones de parámetros para buscar la calibración idónea del programa.

## 1.4 Resumen de la memoria

La estructura de la memoria sigue un orden desde los conceptos más generales y básicos hasta llegar a la experimentación multicriterio final.

Se aborda, en primer lugar, la descripción de los tipos de problemas de secuenciación con optimización multicriterio y los métodos propuestas en la literatura para el análisis de los resultados, donde ya se decide qué métricas se usarán en este trabajo.

A continuación, se concreta en qué tipo de problema se va a resolver, el modelo matemático de la optimización multicriterio de los problemas y el algoritmo que se usará para su resolución.

Tras ello se realiza una breve explicación de las herramientas ya existentes y desarrolladas para llevar a cabo, de una forma más eficiente, la experimentación.

Por último, se realiza una pequeña introducción al nuevo modo de resolución implementado (modo permutation) testeándolo en problemas monobjetivo, y tras dicha introducción se procede a la explicación de la experimentación realizada para la parametrización del algoritmo de la colonia de hormigas en resolución de problemas de secuenciación tipo flowshop.

Al final de la memoria hay un apartado de conclusiones y futuras vías de trabajo donde se presentan, de forma resumida, las valoraciones más importantes de la realización de este trabajo.

# 2. Introducción

## 2.1 Evolución de los procesos productivos

La organización de la producción se ha ido desarrollando a lo largo durante los dos últimos siglos a un ritmo sin precedentes, con grandes cambios como el que se produjo durante la Revolución Industrial pasando de talleres artesanales organizados por gremios a grandes empresas privadas, como la implantación de la producción en cadena, a finales del siglo XIX, a raíz de la corriente del taylorismo en la organización industrial, o, a mitad del siglo XX, con el ejemplo de Toyota en la aplicación de Lean.

Todos estos cambios han ido en la dirección de la masificación de la producción, en primer lugar, y luego en la diversidad de los productos producidos, siempre intentando optimizar recursos para abaratar costes.

Actualmente, la dirección de la organización de la producción va orientada al desarrollo de la Industria 4.0, que busca la automatización, flexibilidad y asignación más eficiente de los recursos.

A la vez que todos estos desarrollos y cambios de filosofía, desde 1950 se comenzaron a estudiar los problemas de 'scheduling', o secuenciación de operaciones, en los que se busca encontrar la secuencia idónea de trabajos en una cadena de operaciones para disminuir tiempos de producción, evitar tiempos ociosos de los recursos disponibles, o, también, buscar la terminación de los trabajos en las fechas acordadas. Gracias al aumento de la capacidad de cálculo computacional, y al desarrollo de ciertos algoritmos de búsqueda de soluciones, se han ido abordando cada vez problemas más variados, más complejos, y, durante los últimos años, buscando una solución a estos optimizando más de un criterio.

## 2.1 Planteamiento del problema

Tal como se define en (Pinedo, 2012) los problemas de scheduling consisten en la búsqueda de la optimización en la asignación de recursos a tareas, en unos periodos de tiempos definidos, con el fin de mejorar uno o varios objetivos. Estos problemas pueden darse tanto en situaciones de optimización de procesos como productivos como en entornos de logística o distribución. En todos ellos se cuenta con una serie de recursos limitados cuyo uso hay que ordenar con el fin de ejecutar una serie de órdenes de trabajo.

El problema tiene su complejidad debido a la gran cantidad de soluciones posibles. Al tratarse de un problema de secuenciación, el número de posibilidades aumenta de forma exponencial cada vez que se añade una tarea nueva y, de esta forma, acaba siendo inabordable una comprobación de cada una de las soluciones para definir cuál es la mejor opción.

Los distintos tipos de problemas con los que nos podemos encontrar están definidos en el Anexo 1, así como las restricciones añadidas que pueden tener estos planteamientos de problemas. De todos ellos, este trabajo estudia flow-shop permutation y flow-shop non-permutation.

Todos estos problemas comparten unas premisas que parten de la siguiente simplificación en la que se define que se van a procesar *N* trabajos en una única máquina, de tal forma que:

- Todos los trabajos están disponibles para ser procesados en t=0.
- La máquina sólo puede procesar un trabajo al mismo tiempo.
- Los tiempos de reconfiguración de máquina o setup entre trabajos son independientes de los trabajos y están incluidos en los tiempos de operación.
- Los descriptores de los trabajos son deterministas y son conocidos de antemano.
- La máquina siempre está disponible (no hay averías).
- La máquina no está ociosa mientras queden trabajos por procesar.
- Cuando una operación finaliza, la siguiente se inicia sin interrupción.

Luego se puede entrar a un nivel superior de detalle y complejidad añadiendo la siguiente información:

- Hay que realizar *N* trabajos, *i=1..N*.
- Se dispone de *M* máquinas para realizar los procesos necesarios, *j=1..M*.
- El proceso de cada trabajo llevado a cabo en cada máquina se llama operación, de manera que la operación del *i*-trabajo realizada en la máquina *j* se denota como $O_{i,j}$.

- Son conocidos los tiempos de proceso de cada una de las operaciones, $p_{i,j}$, y en una primera aproximación éstos son determinísticos.
- Cada máquina solo puede procesar un trabajo al mismo tiempo.
- Cada trabajo solo puede estar siendo procesado por una sola máquina al mismo tiempo.

El principal rasgo de los problemas tipo flow-shop es que las operaciones de cada trabajo tienen un orden establecido para ser procesados en las máquinas. Dentro de ese tipo de problema hay dos variantes de resolución, el modo non-permutation en el que no se tiene por qué respetar la misma secuencia de trabajos en cada máquina, y el modo permutation en el que sí que se tiene que repetir la misma secuencia de trabajos en todas las máquinas.

## 2.2 Valoración de resultados en multiobjetivo

La valoración de los resultados obtenidos en una optimización multiobjetivo requiere un estudio y una concreción de los parámetros mucho más compleja que la optimización monobjetivo.

En el caso de la optimización monobjetivo sólo se necesita comparar directamente resultados (por ejemplo: la configuración de parámetros que obtenga la solución con mejor makespan es la más adecuada para el tipo de problema estudiado). La única dificultad está a la hora de tener en cuenta la aleatoriedad de estos algoritmos y la necesidad de realizar varias iteraciones con cada configuración.

En una optimización multicriterio, hay que valorar qué método seguir para evaluar las soluciones. Existen métodos de ponderación, métodos lexicográficos o métodos a priori. En este trabajo, el algoritmo está diseñado para usar un método a posteriori, que es el de la generación de una frontera de Pareto. Esto supone que la solución a un problema puede ser un conjunto de soluciones, y no solo una, lo cual implica una mayor complejidad a la hora de valorar qué solución es mejor.

Para la creación de dicha frontera el algoritmo compara cada una de las soluciones entre ellas, y se queda con aquellas que no están completamente dominadas por ninguna de las otras. Así, dado un problema de optimización bicriterio, siendo $F_1$ y $F_2$ los criterios, los cuales deben de ser minimizados. Calculadas dos soluciones a ese problema, $x_1$ y $x_2$, si se da que $F_1(x_1) < F_1(x_2)$ y, a la vez, $F_2(x_1) < F_2(x_2)$, se dice que la solución $x_1$ es mejor que $x_2$, o que $x_1$ domina a $x_2$. Pero, si además tenemos una tercera solución $x_3$ con la que se da que $F_1(x_1) < F_1(x_3)$ y, a la vez, $F_2(x_1) > F_2(x_3)$, ¿qué solución es mejor, $x_1$ o $x_3$? Por ello resulta necesario definir distintos tipos de dominancia.

Consideraremos a partir de ahora el operador $\langle$ en la comparación de dos soluciones para el significado de “mejor que”, de tal forma que en el ejemplo anterior podíamos decir que $x_1 \langle x_2$.

Sobre estas notaciones, el trabajo de (Zitzler E. T., 2003) comenzó realizando una extensa definición de todas ellas, y a lo largo de varios trabajos y hasta el último contemplado en la realización de este TFM, que es el de (Zitzler E. K., 2008).

Así, sobre soluciones únicas se puede decir que:

Dominación fuerte (o estricta): se da cuando, para todos los criterios se cumple que $x_1 \langle x_2$. En la Figura 1 se puede ver que la solución $x_1$ domina de forma estricta a $x_2$ y a $x_3$, así como a todas aquellas soluciones que se encuentren dentro del área sombreada.

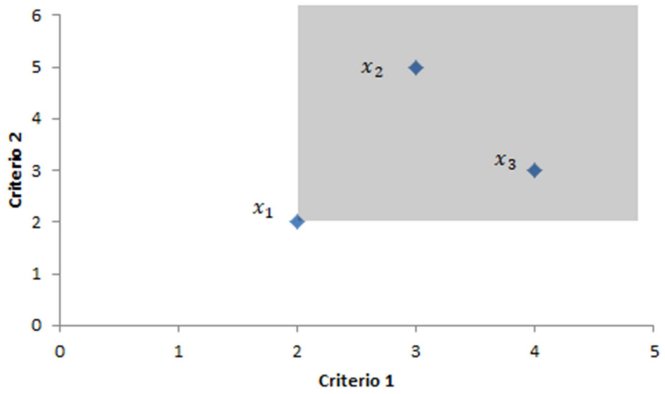



Figura 1: Representación de dominancia fuerte de $x_1$ sobre $x_2$ y $x_3$.

- Dominación débil: si se da este caso se dice que $x_1 \subseteq x_2$, y se da cuando $x_1$ no es peor a $x_2$ en ninguno de los criterios objetivos, pero sí es mejor en alguno de ellos. En la Figura 2 se ve que $x_1$ domina débilmente a $x_2$ y a $x_3$.

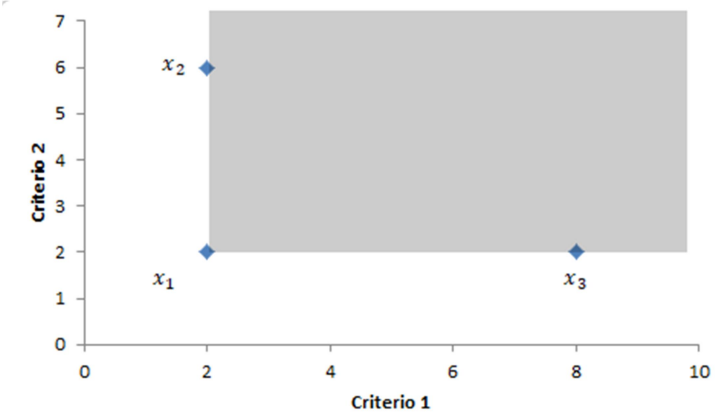



Figura 2: Representación de dominancia débil de $x_1$ sobre $x_2$ y $x_3$.

- Soluciones incomparables: se da cuando no podemos decir que $x_1$ domina débilmente a $x_2$, ni que $x_2$ domina débilmente a $x_1$. En las dos casos anteriores, Figura 1 y Figura 2, se puede decir que $x_2$ y $x_3$ son soluciones incomparables.

Estas definiciones también se aplican a conjuntos de soluciones. Sea $C_1$ y $C_2$ dos conjuntos de soluciones:

- Dominación fuerte (o estricta): El conjunto $C_1$ domina estrictamente a $C_2$ ($C_1 \langle C_2$), si toda solución $x_1$ perteneciente a $C_1$ domina estrictamente a, al menos, una solución $x_2$ perteneciente a $C_2$.
- Dominación débil: El conjunto $C_1$ domina débilmente a $C_2$ ($C_1 \subseteq C_2$) si toda solución $x_1$ perteneciente a $C_1$ domina débilmente a, al menos, una solución $x_2$ perteneciente a $C_2$.
- No comparables: Se da cuando no se cumple que $C_1$ domina débilmente a $C_2$ ni que $C_2$ domina débilmente a $C_1$.

A estos conceptos, se les unen dos definiciones:

- Solución óptima de global de Pareto: Es una solución $x_1$ perteneciente a un conjunto $C_1$ que cumple que ninguna otra solución $x'$ de dicho conjunto la domine estrictamente.
- Conjunto óptimo global de Pareto: Es un conjunto $C'_1$ que pertenece al conjunto de soluciones $C_1$ que está compuesto exclusivamente por todas las soluciones óptimas

globales de Pareto. A este conjunto también se le llama frontera de Pareto. En la Figura 3 los puntos rojos son soluciones óptimas globales del conjunto de Pareto y todos ellos forman la frontera de Pareto.

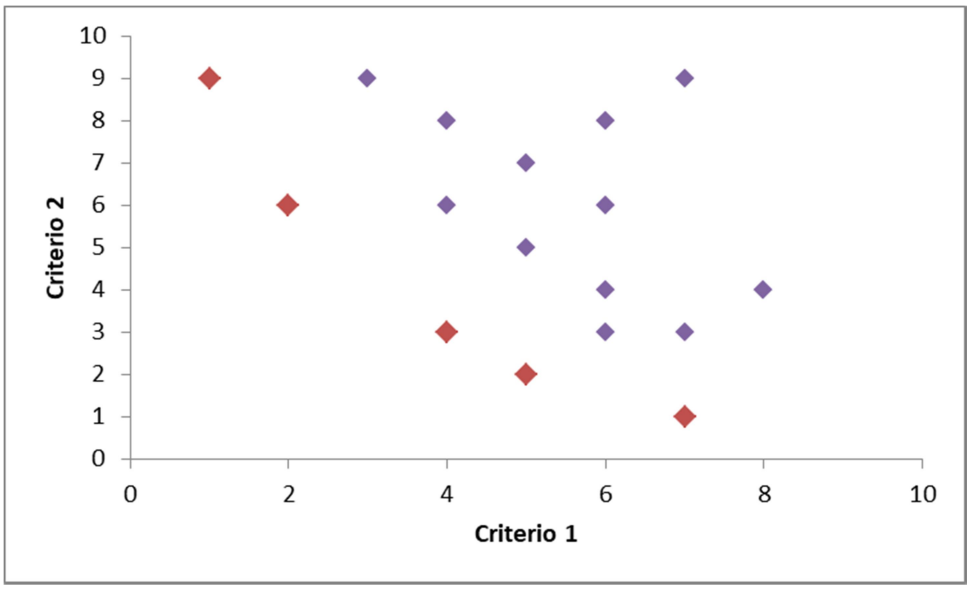



Figura 3: Representación de un conjunto de soluciones donde se aprecia, de color rojo, la frontera de Pareto de dicho conjunto.

En el desarrollo de este trabajo se van a obtener muchas fronteras de Pareto, y la cuestión es establecer algún criterio para valorarlas y decidir cuál es el mejor conjunto de soluciones.

### 2.2.1 Criterios

Antes de entrar a valorar las distintas métricas utilizadas para analizar la optimización de dos criterios hay que conocer en base a qué dos criterios se va a buscar la mejores soluciones.

En general en la literatura de análisis monobjetivo siempre se busca optimizar (minimizar) el makespan $C_{max}$, que corresponde al tiempo neto desde que entra el primer trabajo en la primera máquina hasta que sale el último trabajo de la última máquina. La Figura 4 representa un problema de 3 máquinas y 2 trabajos, en el que este criterio alcanza un valor de 21.

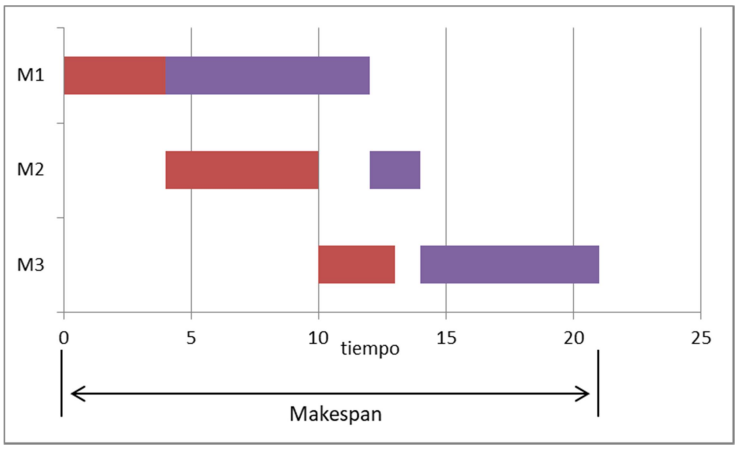



Figura 4: Secuencia de 2 trabajos en 3 máquinas. El makespan es el plazo de tiempo comprendido entre el momento en el que el primer trabajo empieza con su primer proceso hasta que el último trabajo termina en la última máquina.

Para realizar la optimización del makespan solo se requiere, como datos del problema, los tiempos de proceso de cada uno de los trabajos en cada una de las máquinas.

Otro de los criterios es la minimización de la tardanza. Para minimizar dicho criterio se requiere conocer, como dato del problema, las fechas de entrega de cada uno de los trabajos. Con esa información, y tras diseñar una secuencia, se calcula la tardanza como la suma de los tiempos del retraso de cada uno de los trabajos.

Estos, makespan y tardanza, son los dos criterios que se van a optimizar en conjunto para evaluar al algoritmo. En la literatura aparecen otros criterios, como flowtime o adelanto, que también este algoritmo es capaz de optimizar. En el Anexo 2 se describen otros criterios habituales para este tipo de problemas.

### 2.2.2 Métricas

Tras la definición de la creación de una frontera de Pareto y la concreción de los criterios a optimizar, el siguiente punto a determinar es el método seguir para comparar diferentes fronteras de Pareto.

Una de las primeras métricas que se encuentra en la literatura es la de la cobertura, la cual se define como:

$$C(A,B) = \frac{|\{b \in B / \exists a \in A : a \succ b\}|}{|B|}$$

Representa el porcentaje en tanto por uno del número de puntos del conjunto B dominados por el conjunto A. Esta métrica es una buena alternativa para comparar dos soluciones, pero en este trabajo se van a obtener muchas fronteras de Pareto. Esta es la causa que obliga a acudir a otro tipo de métricas que den un valor numérico a cada conjunto de resultados. Estos tipos de indicadores de calidad, estudiados por diversos autores, buscan evaluar con mayor fidelidad las dos características principales que se busca en un frente de Pareto: convergencia y diversidad.

Sea un caso que quiere maximizar dos criterios, como el mostrado en la Figura 5. La convergencia aumenta conforme las soluciones obtenidas se acercan a la supuesta frontera de Pareto óptima; la diversidad es mejor si se obtiene soluciones ampliamente diferentes unas de otras.

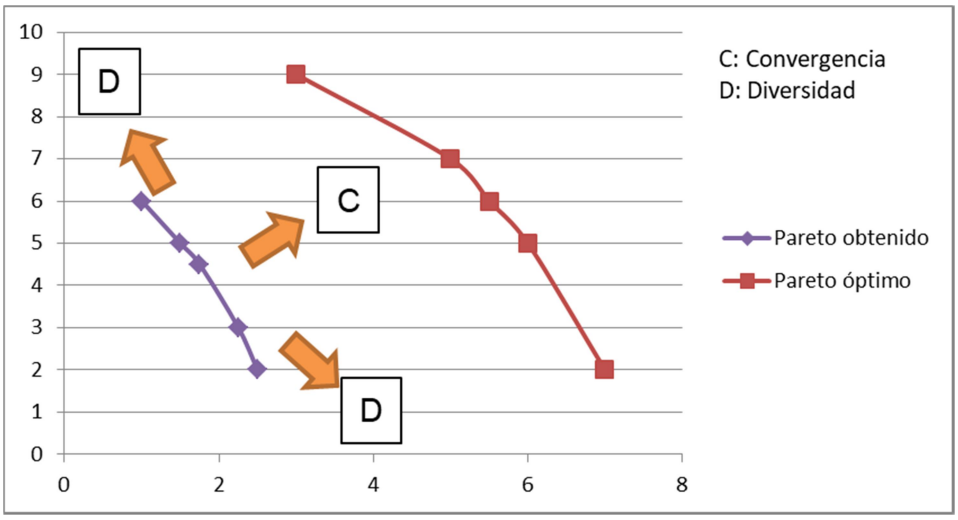



Figura 5: Las dos características principales a valorar de un frente de Pareto para compararlos a otros son su convergencia y su diversidad.

En (Giovanni Lizárraga, 2010) se hace un análisis exhaustivo de las distintas métricas que estudian la diversidad de los frentes de Pareto. Una métrica, en ocasiones utilizada, y que según estos autores no vale, es la de uniformidad. Ésta mide la equidistancia entre los puntos de una frontera de Pareto. Así, teniendo en cuenta las fronteras presentados en la Figura 6, la métrica de uniformidad del conjunto A (rojo) da un mejor resultado que para el conjunto B (morado), lo que no parece muy coherente, todavía más si los dos conjuntos son no dominados entre sí.

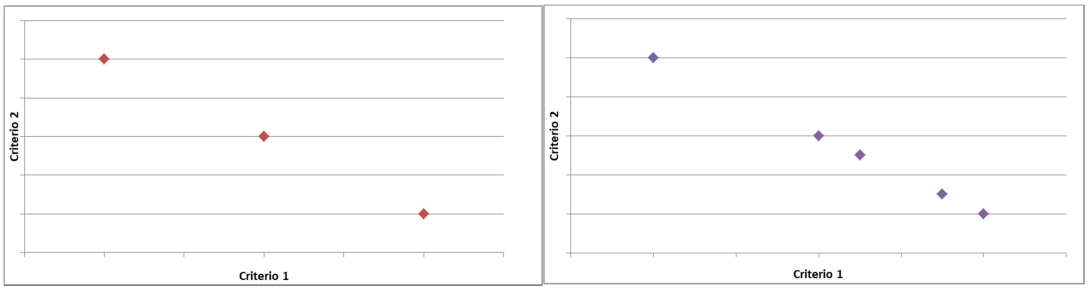



Figura 6: El conjunto A (rojo) da un mejor valor de uniformidad que el conjunto B (morado).

Este indicador no es bueno para nuestro caso ya que carece de monotonía. Esta propiedad se define como "Una métrica I tiene la propiedad de monotonía si al agregar nuevos elementos a un conjunto no dominado A, la valoración del elemento unión es mejor que la de A". Esto significa que, según la Figura 6, el conjunto B (morado) debería tener un mejor valor en la métrica que el conjunto A (rojo) ya que al B se le han añadido dos valores no dominados al conjunto A, y, sin embargo, siguiendo el indicador de uniformidad no se da dicha diferencia.

Otros indicadores que (Giovanni Lizárraga, 2010) también desecha son el número de soluciones que forman parte de la frontera o medir la extensión del conjunto. La cuestión es que estas métricas por sí solas no dan información fiable de la calidad de la frontera de Pareto. Finalmente, en este trabajo, y con el fin de evaluar exclusivamente la diversidad, se crea una métrica con la que se genera un área circular alrededor de cada solución, de forma que dos soluciones muy pegadas no aportan tanta área, por compartir espacio, como una solución alejada del resto.

La métrica EAF, propuesta inicialmente por (Grunert da Fonseca, 2001) y desarrollada por (López-Ibáñez, 2006), se basa en porcentajes de aparición en áreas de la gráfica. El problema es que requiere muchas iteraciones y guardar todos los resultados, no sólo las fronteras de Pareto. Para nuestro trabajo, es una métrica inviable ya que requiere muchas iteraciones y además supone un gran tiempo computacional su cálculo.

Dada la necesidad de valorar tanto la convergencia como la diversidad, se van a calcular dos métricas $I_H$ e $I_\varepsilon$ propuestas, respectivamente, en (Zitzler E. , 1999) y (Zitzler E. T., 2003)

La primera $\mathrm{I_H}$ es el indicador unario de la métrica hipervolumen. El hipervolumen mide el área que cubren las soluciones de una frontera de Pareto, teniendo unos valores de referencia límite $S_1$ y $S_2$ para calcular dichas secciones. En la Figura 7 se representa dicha área.

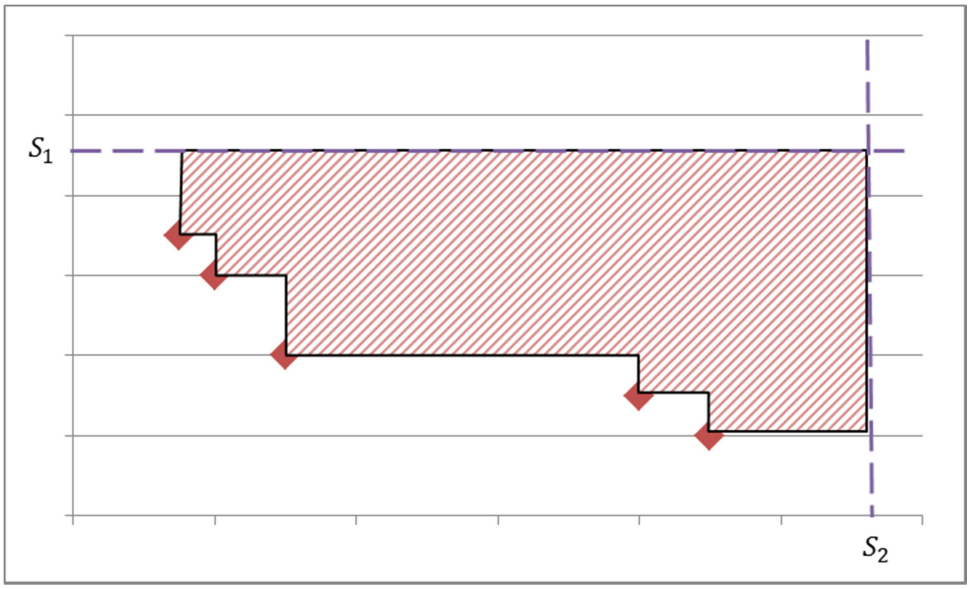



Figura 7: Frente de Pareto y representación del hipervolumen.

El hecho de que sea un indicador unario es que es una métrica escalada y normalizada. Para ello, se divide el valor del área entre una referencia usada en el resto de soluciones. Esta referencia es el valor del hipervolumen del mejor frente de Pareto conocido. Siguiendo un criterio parecido al presentado en (Minella, 2014), se establece como valores de referencia, $S_1$ y $S_2$, los valores máximos de la frontera de Pareto óptima -de cada uno de los dos criterios multiplicados por 3. Para normalizar, se divide entre el hipervolumen de la mejor solución conocida. De esta forma, cuanto mayor es el valor de $I_H$ mejor es el frente de Pareto.

El indicador epsilon, $I_\varepsilon$, es el mayor factor por el que hay que dividir cada una de las soluciones de un conjunto A de forma que éste siga siendo débilmente dominado por B, siendo B el mejor conjunto de soluciones para ese problema. Así, si épsilon toma el valor de 1, supone que alguno de los frentes de Pareto conseguidos comparte solución con los Paretos óptimos conocidos; cuanto mayor sea el valor del indicador, el conjunto es peor. Si épsilon toma un valor menor de 1, es que se ha conseguido un conjunto que mejora el mejor encontrado hasta el momento.

En este trabajo, el mejor conjunto de soluciones está formado por las soluciones aportadas por (Minella, 2014), ya que resuelve, por otros procedimientos, los mismos problemas.

## 2.3 Métodos de resolución

Para la resolución de los problemas de secuenciación de operaciones existen diversidad de algoritmos que se pueden clasificar en los siguientes grupos:

- Métodos exactos: son los procedimientos que buscan la solución óptima del problema, aunque en general no son suficientemente robustos y para problemas grandes requiere un tiempo de cálculo excesivo.
- Métodos heurísticos: también llamados métodos iterativos de aproximación, consisten en realizar una búsqueda de una solución partiendo de su mejor resultado conocido. Estos procedimientos no aseguran obtener la solución óptima.
- Algoritmos metaheurísticos: son una evolución de los heurísticos, además de trabajar con la información de la mejor solución obtenida tienen procedimientos para hacer una búsqueda más global con menos riesgo a converger demasiado rápido en una solución. Son los algoritmos más utilizados porque obtienen buenos resultados con cualquier problema por complejo que sea.

El algoritmo que utilizaremos en este trabajo es uno metaheurístico basado en una colonia de hormigas.

En el Anexo 3 hay una descripción más detallada con ejemplos concretos de algoritmos de cada una de estas clasificaciones.

# 3. Caracterización del problema resuelto

## 3.1 Descripción

Tal y como se ha comentado anteriormente el problema que abordaremos en las experimentaciones se trata de un problema de tipo flow shop (es decir, que el orden de las máquinas es el mismo en todos los trabajos y debe respetarse), el cual resolveremos en modo permutation y non-permutation.

En modo non-permutation la secuencia de trabajos realizados en una máquina no tiene por qué repetirse en el resto de máquinas (ver Figura 8) y, de esta forma, las combinaciones posibles de soluciones son $(n!)^m$, siendo *n* el número de trabajos y *m* el número de máquinas.

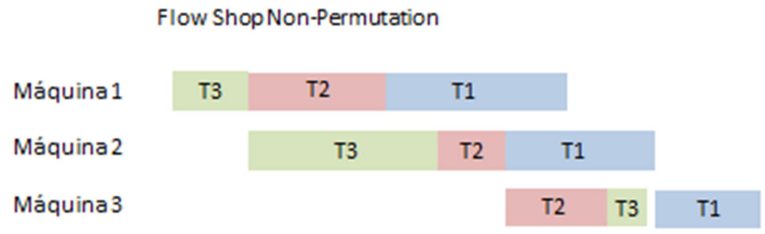



Figura 8: Secuencia resuelta en modo non-permutation. Mientras que en las máquinas 1 y 2 el orden de los trabajos es T3, T2 y T1, en la máquina 3, con el fin de tener un menor makespan, cambia el orden de los trabajos.

Mientras que en el caso de permutation se debe respetar la misma secuencia de trabajos en todas las máquinas (ver Figura 9) y por tanto, las combinaciones posibles son $(n!)$.

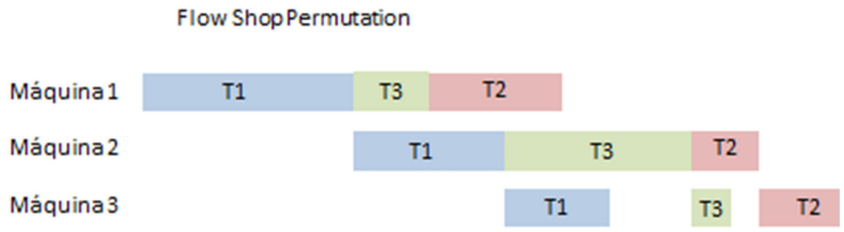



Figura 9: Secuencia resuelta en modo permutation, donde se observa que el orden de los trabajos en cada máquina es el mismo.

La resolución en modo non-permutation es poco común dentro de la literatura, justificando principalmente que el modo permutation es un caso más cercano a la realidad.

Sin embargo uno de los objetivos principales de este trabajo será evaluar el funcionamiento del algoritmo de optimización en ambos casos, ya que se quiere comprobar si el hecho de tener más soluciones disponibles (en el modo non-permutation), y entre ellas las soluciones del modo permutation (ver Figura 10), supone conseguir mejores o peores resultados en la optimización.

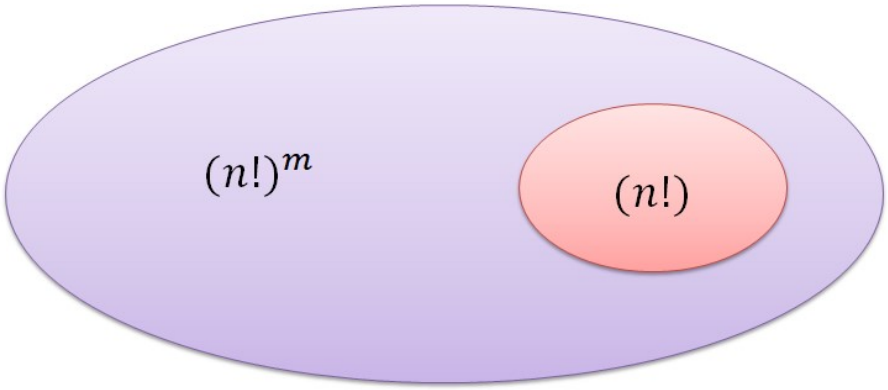



Figura 10: Representación del número de soluciones que se pueden obtener mediante el modo non-permutation (morado) y el modo permutation (rojo).

Además, el programa tiene implementado el algoritmo de forma que puede aceptar otras consideraciones entre las características del problema, como diferentes tamaños de lote, la posibilidad de que un trabajo se salte una máquina si no tiene tiempo de proceso en esta y otras casuísticas que se pueden dar en un caso real. Al no tratar problemas tan complejos en este trabajo se deja la descripción de estos otros elementos en el Anexo 4. A su vez, y por no haber sido tan fundamental el análisis de éste en el presente trabajo, se refleja en el Anexo 5 el modelo matemático y la formulación de la programación empleada para la resolución de estos problemas.

## 3.2 Algoritmo de resolución, ACO

Entre 1940 y 1950 Pierre-Paul Grassé observó que las termitas, de forma individual, son limitadas en cuanto a la búsqueda de comida, sin embargo, en una colonia, el comportamiento de una de ellas afectaba al resto mediante las feromonas que segregan en su trayecto. En (Dorigo, Optimization, Learning and Natural Algorithms, 1992) aplicó este modelo en problemas de combinatoria, desarrollando dos fórmulas básicas en el algoritmo. En (Dorigo, 2006) se presentan los aspectos básicos de este algoritmo.

Buscando emular el comportamiento de las hormigas a la hora de decidir el camino a seguir se define la ecuación:

$$p_{xy}^{k} = \frac{\tau_{xy}^{\alpha}\eta_{xy}^{\beta}}{\Sigma(\tau_{xy}^{\alpha}\eta_{xy}^{\beta})} \qquad (1)$$

Siendo $p_{xy}^{k}$ la probabilidad de que la hormiga $k$ coja el camino que va desde $x$ hasta $y$. En el segundo miembro de la ecuación se encuentran valores de feromona y heurística para dar mayor peso o menos a cada camino.

Y, por otro lado, buscando representar la evaporación de la feromona de las hormigas, se usa:

$$\tau_{xy} \leftarrow (1-\rho)\cdot\tau_{xy} + \sum_{k}\Delta\tau_{xy}^{k} \qquad (2)$$

Siendo $\rho$ el coeficiente de evaporación de las feromonas y $\Delta\tau_{xy}^{k}$ la cantidad de feromonas depositadas por la hormiga k-ésima.

## 3.3 Parámetros de MACO

Partiendo de la base explicada en el punto anterior se desarrollan los algoritmos basados en las colonias de hormigas añadiendo más actualizaciones de la feromona a mitad de las

iteraciones, dando aleatoriedad al algoritmo a regirse por la probabilidad de escoger un camino o, directamente, ir siempre al que más probabilidad tenga, o añadiendo complementos ajenos por completo al ACO original como la aplicación de la búsqueda de mejoras locales mediante permutación de órdenes de trabajo.

Todas estas herramientas están parametrizadas y afectarán en mayor o menor medida en función de la configuración de parámetros que diseñemos. Y, tal y como se vio en (Gutiérrez, 2016), dicha calibración resultó ser fundamental para la obtención de unos buenos resultados en la optimización monobjetivo.

A continuación se hace una pequeña explicación de los parámetros que definen las funciones que caracterizan al algoritmo.

- Definición de secuencia inicial, $Sec_O$: el algoritmo comienza elaborando una secuencia inicial con la que empezará a realizar sus cálculos. Para la definición de esa primera solución se elegirá de manera secuencial una operación de entre todas las factibles. Se define el conjunto de factibles como el conjunto de operaciones, una por trabajo, que sea la que deba realizarse a continuación. Para seleccionar una entre las factibles se aplican criterios de tipo heurístico: menor tiempo de proceso, mayor número de operaciones restantes en el trabajo y número de operaciones restantes en máquina.

  Establecida $Sec_O$, ya se pueden calcular los objetivos de dicha secuencia, y en este caso, directamente entra a formar parte de la frontera de Pareto.

- Calcular feromona inicial: teniendo ya definida esa primera secuencia, se calcula la feromona para cada objetivo *s*, que no es más que la información depositada por la hormiga al haber programado la orden o2 tras la orden o1.

$$\tau_0^s = \tau_{o1\,o2}^s = \frac{1}{\text{nº órdenes} * OBJ_s^{Sec\_0}} \tag{3}$$

- Definición de secuencia siguiente: función donde se condensa gran parte del algoritmo. El procedimiento secuencial consiste en ir definiendo el conjunto de órdenes factibles (aquellas que pueden ser programas cumpliendo restricciones) y seleccionar una de este conjunto, hasta que todas las órdenes hayan sido seleccionadas. Para elegir una de las órdenes del conjunto se aplica la regla pseudoaleatoria, que se basa en la existencia de un parámetro q0 con valor entre 0 y 1, de manera que cada vez que se aplica esta regla se genera un número aleatorio, q, entre 0 y 1. Así, la probabilidad de que tras haber programado la orden o1, se programa la orden o2.

$$p_{o1\,o2} = \begin{cases} si\ q > q0 & \begin{cases} \frac{\prod \tau_{\mathrm{o1\,o2\,s}}^{\alpha} \cdot \eta_{\mathrm{o1\,o2}}^{\beta}}{\sum_{o2 \in ConjFact} \prod \tau_{\mathrm{o1\,o2\,s}}^{\alpha} \cdot \eta_{\mathrm{o1\,o2}}^{\beta}} & \text{si } \mathrm{o2} \in \mathrm{ConjFact(o1)} \\ 0 \quad en\ otro\ caso & \prod \tau_{\mathrm{o1\,o2\,s}}^{\alpha} \end{cases} \\ si\ q \leq q0 & \begin{cases} 1 \quad \text{si } \mathrm{o2} = \arg\max \prod_{o2 \in ConjFact} \tau_{\mathrm{o1\,o2\,s}}^{\alpha} \cdot \eta_{\mathrm{o1\,o2}}^{\beta} \\ 0 \quad en\ otro\ caso & \prod \tau_{\mathrm{o1\,o2\,s}}^{\alpha} \end{cases} \end{cases} \tag{4}$$

- Actualizar feromona en línea: conforme se van programando las órdenes o bien una vez finalizada la solución, se actualiza la feromona, para representar el hecho de que una hormiga ha transitado por el arco que une o1 con o2.

$$\tau_{\mathrm{o1\,o2}}^{\mathrm{s}} = \tau_{\mathrm{o1\,o2}}^{\mathrm{s}} \cdot (1 - \varphi) + \tau_0^{\mathrm{s}} \cdot \varphi \tag{5}$$

- Actualizar feromona a posteriori: en algunos casos, una vez que se han generado todas las soluciones de una iteración, se modifica la feromona de los arcos que pertenecen a las mejores soluciones ($S_{MG}$). Habitualmente se evapora o disminuye feromona para que de esta manera el algoritmo no se quede atascado el óptimos locales. En este caso, hay que prestar especial atención a SMG, ya que depende del objetivo con que se evalúe. Por ello, las soluciones se ordenan según el valor de cada criterio, y se aplica a las t-mejores soluciones según cada criterio la actualización de la feromona que corresponde al criterio en que son buenas.

$$\tau^{s}_{o1\,o2} = \tau^{s}_{o1\,o2}.(1-\rho) + \rho.f\left(OBJ_S(S_{MG})\right) \quad \forall a_{o1\,o2} \in S_{MG} \qquad (6)$$

- Limitar valor de feromona: una vez se ha actualizado el valor de la feromona de cada criterio y arcos, se revisa que ésta no esté por debajo ni por encima de unos límites. Esta técnica se llama min-max y con ella, el valor de la feromona está limitado en un rango $[\tau^{s}_{min}, \tau^{s}_{max}]$ y se evita que haya arcos tanto demasiado (mucha feromona) como poco apetecibles.
- Mejora local: el algoritmo de hormigas tiene una buena estrategia para diversificar, es decir, para recorrer el espacio de factibilidad, pero no es especialmente bueno en la intensificación o revisión de soluciones en el entorno de una ya generada. Por ello, se aplica técnicas de mejora local a un conjunto de las mejores soluciones, teniendo en cuenta para ello las definiciones de conjunto y criterio vistas para la actualización de feromona a posteriori. La mejora se basa en la técnica de intercambio de dos órdenes.
- Resetear feromona: tras un número determinado de iteraciones o tiempo de computación, si la frontera de Pareto no ha sido modificada, se entiende que el algoritmo se ha quedado estancado. Para evitar este problema, tras estos límites, se resetea la feromona, es decir se determina una nueva solución inicial teniendo en cuenta las soluciones del frente de Pareto, y a partir de los objetivos de ésta, se calcula el valor de las nuevas feromonas iniciales para cada criterio, actualizando la feromona de todos los arcos a este valor.

Los parámetros del algoritmo de hormigas cuyo valor hay que definir, independientemente del enunciado concreto a resolver, son los siguientes:

- Tiempo máximo de simulación: corresponde al tiempo máximo de computación medido en segundos que el programa puede estar calculando.
- Iteraciones máximas de simulación: corresponde al número de iteraciones máximas que puede realizar el programa.
- Número de hormigas: corresponde al número de soluciones generadas en cada iteración.
- Heurísticas: se numeran desde 0 hasta 10 y se pueden añadir al programa tantas como se quieran, y sirven, (definido como valor $\eta$) para dar mayor o menor prioridad a una operación de un proceso en función de sus características.

  Las heurísticas que se han programado son:

  Heur00: operación que empiece antes (EST).

  Heur01: operación que termine antes (EFT).

  Heur02: operación que pertenezca a un trabajo con menor tiempo de proceso (SPT).

  Heur03: operación que pertenezca a un trabajo con mayor tiempo de proceso (LPT).

  Heur04: tiempo de proceso menor respecto al tp restante de trabajo (LWR).

  Heur05: tiempo de proceso mayor respecto al tp restante de trabajo (MWR).

  Heur06: tiempo de proceso menor respecto al tp total de trabajo (LTW).

  Heur07; tiempo de proceso mayor respecto al tp total de trabajo (MTW).

Heur08: prisa por acabar el trabajo (Límite superior - tiempo fin orden - tiempo restante trabajo).

Heur09: no prisa por acabar el trabajo (Límite inferior - tiempo fin orden - tiempo restante trabajo).

Heur10: esta heurística representa que no se va a aplicar ninguna.

- q0: parámetro que determina la forma de seleccionar una operación del conjunto de factibles (regla pseudoaleatoria). Este parámetro siempre vale entre 0 y 1, pero puede tener un valor fijo a lo largo de todo el cálculo o ir variando. Es importante tener en cuenta que cuanto menor sea este valor, más probable será seleccionar la orden cuyo producto de información heurística con feromona sea mayor, es decir, el proceso de elección estará más dirigido y la aleatoriedad será menor.
- α: corresponde al exponente de la feromona en la regla pseudoaleatoria. Este parámetro puede ser el mismo para todas las feromonas o distinto.
- β: corresponde al exponente sobre la heurística en la regla pseudoaleatoria. Este parámetro puede ser el mismo para todas las heurísticas o distinto.
- Actualización de feromona en línea: en este caso hay que elegir si se da el caso, y además la forma, paso a paso conforme se seleccionan las órdenes o una vez establecida la solución completa.
- φ: coeficiente de actualización de feromona en línea. Este parámetro vale entre 0 y 1, puede ser fijo o variable a lo largo del cálculo.
- Actualización a posteriori: hay cuatro tipos y la posibilidad de no elegir ninguna, y además hay que elegir la variable que corresponde al número de soluciones a las que se le aplica dicha actualización. La forma de codificar es Tipo_NumSol

  0_NS: no se hace actualización a posteriori.

  1_NS: las NS-mejores soluciones de la iteración actualizan la feromona correspondiente al objetivo en que son buenas.

  2_NS: las NS-mejores soluciones según un objetivo de la iteración actualizan todas las feromonas (si son todas, el nº de soluciones será -1).

  3_NS: las NS-mejores soluciones del Pareto según un objetivo actualizan la feromona correspondiente al objetivo en que son buenas.

  4_NS: las NS-mejores soluciones del Pareto según un objetivo actualizan todas las feromonas (si son todas, el nº de soluciones será -1).

- ρ: coeficiente de actualización de feromona a posteriori. Este parámetro vale entre 0 y 1, puede ser fijo o variable a lo largo del cálculo.
- Minmax: puede estar activo o no, en caso de estarlo limita los valores de la feromona de las órdenes para que no sean muy superiores o inferiores respecto a las del resto.
- Reset: en caso de estar activo elegimos el número iteraciones que pueden pasar sin encontrar una solución mejor, y una vez pasado ese límite se reseteará la información que da la feromona.
- Mejora local: se puede elegir si utilizar este complemento o no, y en caso de hacerlo podemos elegir a cuantas soluciones aplicarla. Consiste en analizar las n-mejores secuencias y buscar pequeños cambios en el orden de las operaciones para lograr un mejor resultado.

En total hay 15 parámetros/opciones que será necesario darles un valor. Como se puede ver, el número es muy elevado, además de que el número de valores que pueden tomar también lo es.

# 4. Adecuación del algoritmo MACO y diseño de herramientas para su análisis

Durante (Gutiérrez, 2016) se desarrollaron nuevas funciones y herramientas para el algoritmo MACO, orientadas principalmente a la generación de baterías de experimentos más grandes y a la obtención de resultados en un formato que fuesen más sencillos de analizar. En este trabajo se ha buscado dar un paso más en la comprensión del funcionamiento del algoritmo y, por otra parte, la optimización multicriterio requiere una automatización del cálculo de las métricas anteriormente descritas.

Por ello se han desarrollado las herramientas descritas a continuación.

## 4.1 Situación de partida

Al inicio de (Gutiérrez, 2016), se disponía de un algoritmo en C++, en el que estaba implementado un ACO multiobjetivo donde los parámetros se introducían mediante un fichero de texto y, este algoritmo era únicamente capaz de realizar una sola iteración.

Debido a la necesidad de realizar una gran cantidad de experimentos, con muchas configuraciones de parámetros distintas y varias réplicas de cada una de esas configuraciones, se diseñó y programó una interfaz (ver Figura 11). Además, se programaron una serie de funciones con las que, de una sola vez, se podía diseñar una batería de simulaciones. Así, el algoritmo podía funcionar durante largos periodos de tiempo seguidos sin que fuese necesario estar delante del ordenador. Además, se modificó la salida de los resultados para que fuese más sencilla su importación en un archivo Excel en el que luego, de forma manual y mediante mapas de color, se realizaba el análisis monobjetivo.

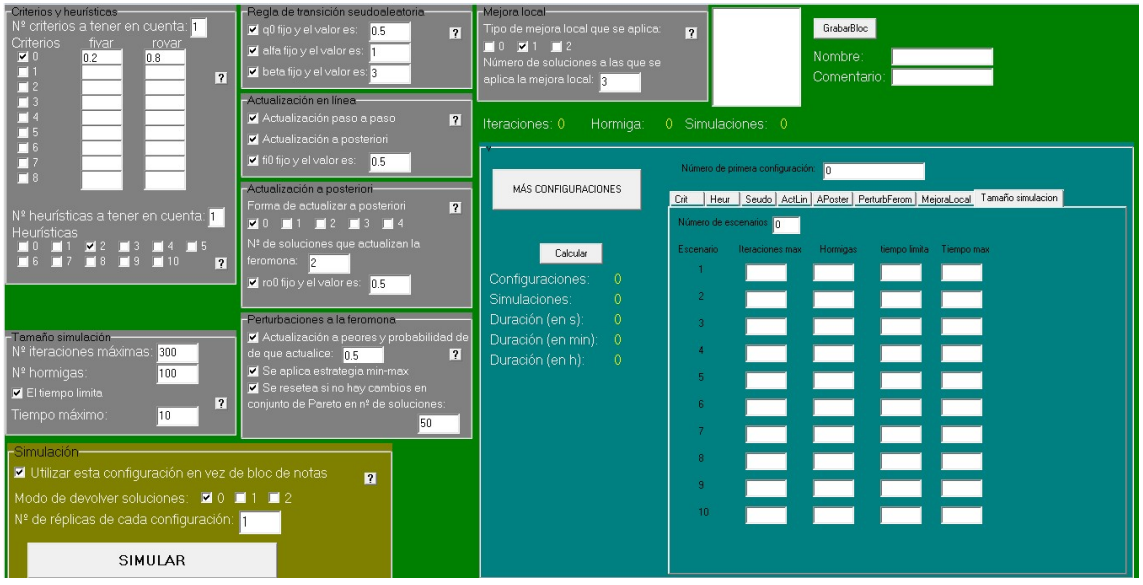



Figura 11: Captura de pantalla de la interfaz creada para la generación de diversas baterías de configuraciones.

## 4.2 Herramienta para análisis de resultados

Para realizar un análisis más rápido y efectivo de la optimización multicriterio, en este trabajo se ha diseñado y programado una herramienta en el entorno Excel que facilite la obtención de las conclusiones del análisis del algoritmo.

Esta herramienta, programada en Visual Basic para aplicaciones, extrae los resultados en el formato que los devuelve el algoritmo, los clasifica, les da un valor siguiendo las métricas

propuestas, y, con el formato de un *dashboard interactivo*, visualiza las fronteras de Pareto de las mejores soluciones y muestra los parámetros con los que se obtienen las mejores fronteras.

A continuación se muestran las principales características de este dashboard. Una explicación más extensa a modo manual de instrucciones y otra más técnica a nivel de cómo se ha diseñado la programación se puede encontrar en Anexo 6 y Anexo 7.

El dashboard permite comparar resultados entre distintas configuraciones (o conjuntos de configuraciones), pudiendo analizar dos casos a la vez, el azul y el naranja, de tal forma que, seleccionando los parámetros que queramos en los filtros, se modificarán los resultados reflejados en los recuadros.

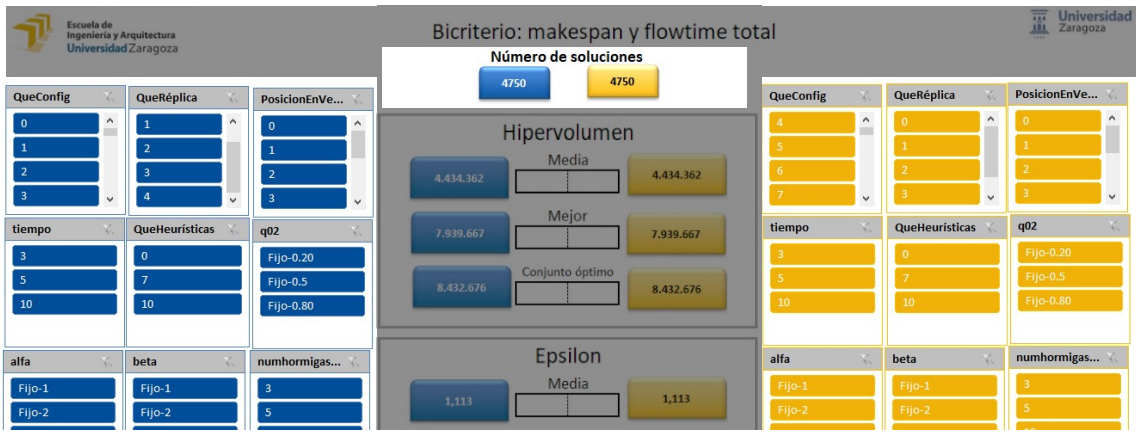



Figura 12: En los laterales se muestra los filtros disponibles para elegir qué conjuntos de soluciones reflejar. Arriba aparece el número de soluciones que han sido obtenidas con los filtros seleccionados en los laterales.

En la Figura 12 se observa el Excel previo a pulsar ningún filtro, y en la sección azul de la Figura 13 el resultado de visualizar solo las configuraciones con tiempo de computación de 10 segundos y con un valor de q0 igual a 0,2 (filtros). Se ve cómo, además de otros parámetros, ha cambiado el número de soluciones, ya que, entre todas las configuraciones realizadas con esos filtros se han obtenido 600 soluciones. El número de soluciones obtenidas en todas las configuraciones es 4750, tal y como aparece en el recuadro azul de la Figura 12, donde no se ha activado filtro alguno.

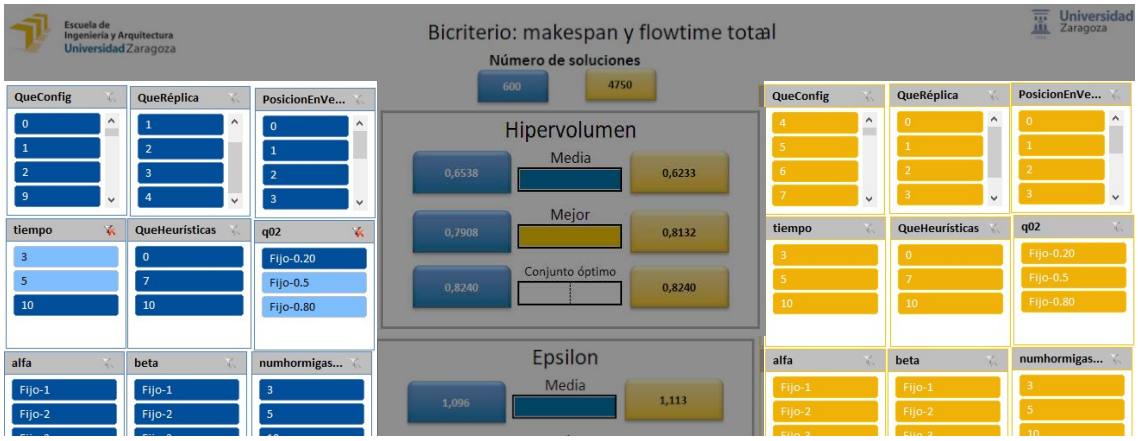



Figura 13: Dashboard tras hacer una selección, en el caso azul, eligiendo las experimentaciones con un tiempo de 10 segundos y un valor de q0=0.2

Además, también se recalculan, con las soluciones seleccionadas, las métricas de hipervolumen y de épsilon. Así, para cada conjunto de soluciones, se muestra la media del

resultado de cada una de las métricas, el valor de la mejor configuración, y el valor de dicha métrica haciendo una frontera de Pareto óptima con todas las soluciones de este conjunto de configuraciones.

En la Figura 14 se ve cómo este subconjunto configuraciones (tiempo de computación de 10 s y q0=0.2), mostradas en el lado azul, tienen mejor media de hipervolumen que todas las configuraciones, lado naranja. También se observa que la mejor solución es compartida para el caso azul y naranja, por lo que dicha solución se obtiene con los parámetros del filtro. Además, la frontera de Pareto óptima es mejor en el caso naranja, resultado que era esperable, ya que contiene más soluciones y, entre ellas, todas las soluciones azules.

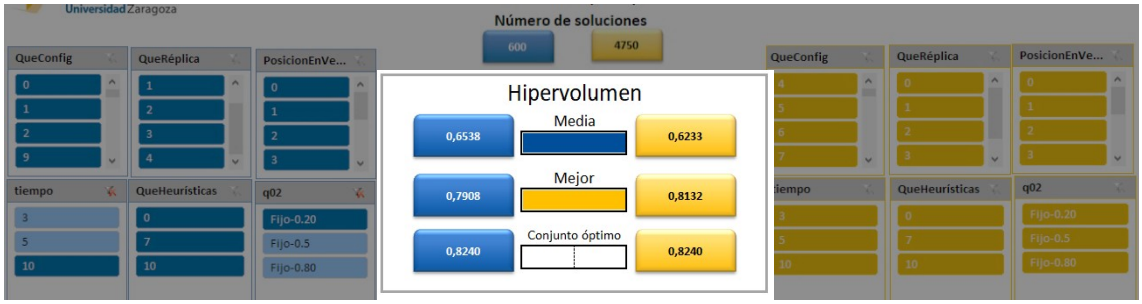



Figura 14: Muestra de los valores de hipervolumen con los filtros seleccionados.

De la misma manera, en la Figura 15 se presenta el mismo esquema de cálculo pero ahora con la métrica Epsilon.

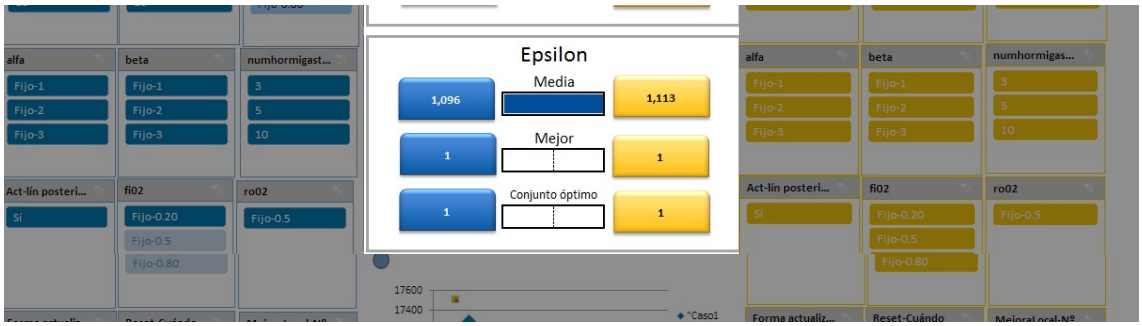



Figura 15: Muestra de los valores de épsilo con los filtros seleccionados.

En la Figura 16 se ve la frontera de Pareto del conjunto azul, que comparte todos los puntos con el caso naranja salvo dos. Además, la frontera de Pareto óptima naranja es la misma que la mejor conocida, ya que no se ha pulsado ningún filtro, y, por definición, ambas fronteras son las mismas.

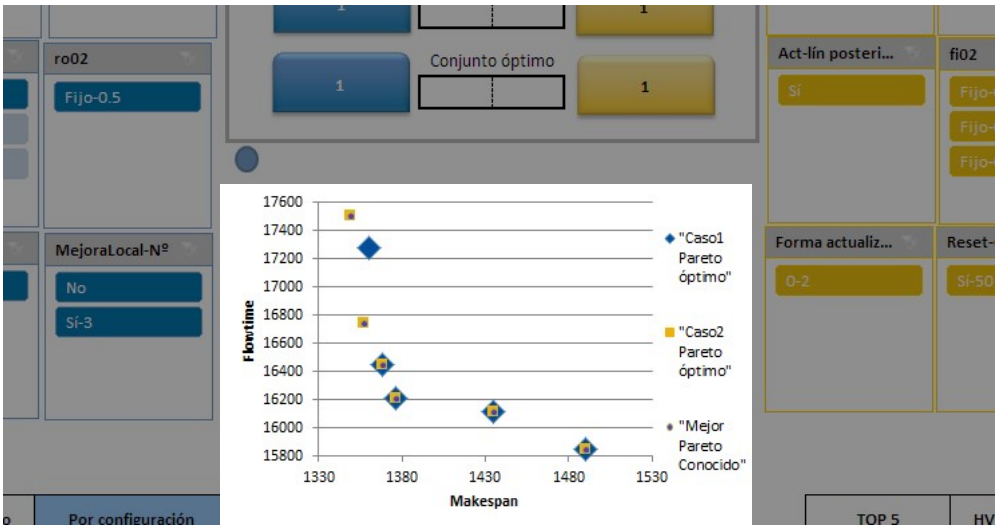
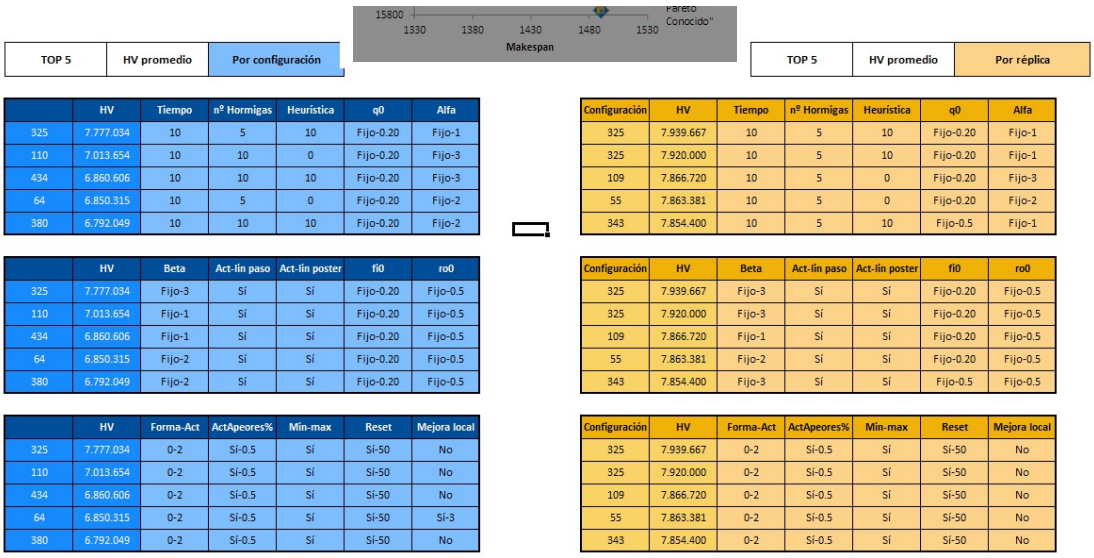



Figura 16: Representación de las fronteras de Pareto de ambos casos y, en morado, las mejores soluciones obtenidas de toda la experimentación independientemente de los filtros escogidos.

La herramienta también da la posibilidad de generar la frontera de Pareto óptima (puntos morados en Figura 16) con resultados introducidos a mano que no necesariamente hayan sido obtenidos en la experimentación que estamos estudiando.

Por último, también se presentan, tal y como se puede ver en la Figura 17, las mejores configuraciones y réplicas, según la métrica elegida, así como el valor de los parámetros con los que se ha conseguido estos resultados.




| | HV | Tiempo | nº Hormigas | Heurística | q0 | Alfa |
|---|---|---|---|---|---|---|
| 325 | 7.777.034 | 10 | 5 | 10 | Fijo-0.20 | Fijo-1 |
| 110 | 7.013.654 | 10 | 10 | 0 | Fijo-0.20 | Fijo-3 |
| 434 | 6.860.606 | 10 | 10 | 10 | Fijo-0.20 | Fijo-3 |
| 64 | 6.850.315 | 10 | 5 | 0 | Fijo-0.20 | Fijo-2 |
| 380 | 6.792.049 | 10 | 10 | 10 | Fijo-0.20 | Fijo-2 |

| Configuración | HV | Tiempo | nº Hormigas | Heurística | q0 | Alfa |
|---|---|---|---|---|---|---|
| 325 | 7.939.667 | 10 | 5 | 10 | Fijo-0.20 | Fijo-1 |
| 325 | 7.920.000 | 10 | 5 | 10 | Fijo-0.20 | Fijo-1 |
| 109 | 7.866.720 | 10 | 5 | 0 | Fijo-0.20 | Fijo-3 |
| 55 | 7.863.381 | 10 | 5 | 0 | Fijo-0.20 | Fijo-2 |
| 343 | 7.854.400 | 10 | 5 | 10 | Fijo-0.5 | Fijo-1 |

| | HV | Beta | Act-lín paso | Act-lín poster | fi0 | ro0 |
|---|---|---|---|---|---|---|
| 325 | 7.777.034 | Fijo-3 | Sí | Sí | Fijo-0.20 | Fijo-0.5 |
| 110 | 7.013.654 | Fijo-1 | Sí | Sí | Fijo-0.20 | Fijo-0.5 |
| 434 | 6.860.606 | Fijo-1 | Sí | Sí | Fijo-0.20 | Fijo-0.5 |
| 64 | 6.850.315 | Fijo-2 | Sí | Sí | Fijo-0.20 | Fijo-0.5 |
| 380 | 6.792.049 | Fijo-2 | Sí | Sí | Fijo-0.20 | Fijo-0.5 |

| Configuración | HV | Beta | Act-lín paso | Act-lín poster | fi0 | ro0 |
|---|---|---|---|---|---|---|
| 325 | 7.939.667 | Fijo-3 | Sí | Sí | Fijo-0.20 | Fijo-0.5 |
| 325 | 7.920.000 | Fijo-3 | Sí | Sí | Fijo-0.20 | Fijo-0.5 |
| 109 | 7.866.720 | Fijo-1 | Sí | Sí | Fijo-0.20 | Fijo-0.5 |
| 55 | 7.863.381 | Fijo-2 | Sí | Sí | Fijo-0.20 | Fijo-0.5 |
| 343 | 7.854.400 | Fijo-3 | Sí | Sí | Fijo-0.5 | Fijo-0.5 |

| | HV | Forma-Act | ActApeores% | Min-max | Reset | Mejora local |
|---|---|---|---|---|---|---|
| 325 | 7.777.034 | 0-2 | Sí-0.5 | Sí | Sí-50 | No |
| 110 | 7.013.654 | 0-2 | Sí-0.5 | Sí | Sí-50 | No |
| 434 | 6.860.606 | 0-2 | Sí-0.5 | Sí | Sí-50 | No |
| 64 | 6.850.315 | 0-2 | Sí-0.5 | Sí | Sí-50 | Sí-3 |
| 380 | 6.792.049 | 0-2 | Sí-0.5 | Sí | Sí-50 | No |

| Configuración | HV | Forma-Act | ActApeores% | Min-max | Reset | Mejora local |
|---|---|---|---|---|---|---|
| 325 | 7.939.667 | 0-2 | Sí-0.5 | Sí | Sí-50 | No |
| 325 | 7.920.000 | 0-2 | Sí-0.5 | Sí | Sí-50 | No |
| 109 | 7.866.720 | 0-2 | Sí-0.5 | Sí | Sí-50 | No |
| 55 | 7.863.381 | 0-2 | Sí-0.5 | Sí | Sí-50 | No |
| 343 | 7.854.400 | 0-2 | Sí-0.5 | Sí | Sí-50 | No |

Figura 17: Imagen del dashboard donde se muestran los valores de los parámetros en las configuraciones con mejores resultados de hipervolumen.

Este top de resultados se realiza teniendo en cuenta también los filtros seleccionados descritos anteriormente. De ahí que, en el caso azul, se muestran las 5 configuraciones que tienen mejor media calculando el hipervolumen con los filtros activados.

Además, junto a la realización de esta herramienta para el análisis de optimizaciones multiobjetivo se ha desarrollado también la posibilidad de evaluar los resultados de una optimización monobjetivo. Se sigue la misma filosofía, trabajando con dos casos distintos, y,

además, en esta ocasión se visualiza también una gráfica de barras, visualizando valores medios y desviaciones, del criterio a optimizar, tal y como se muestra en la Figura 18:

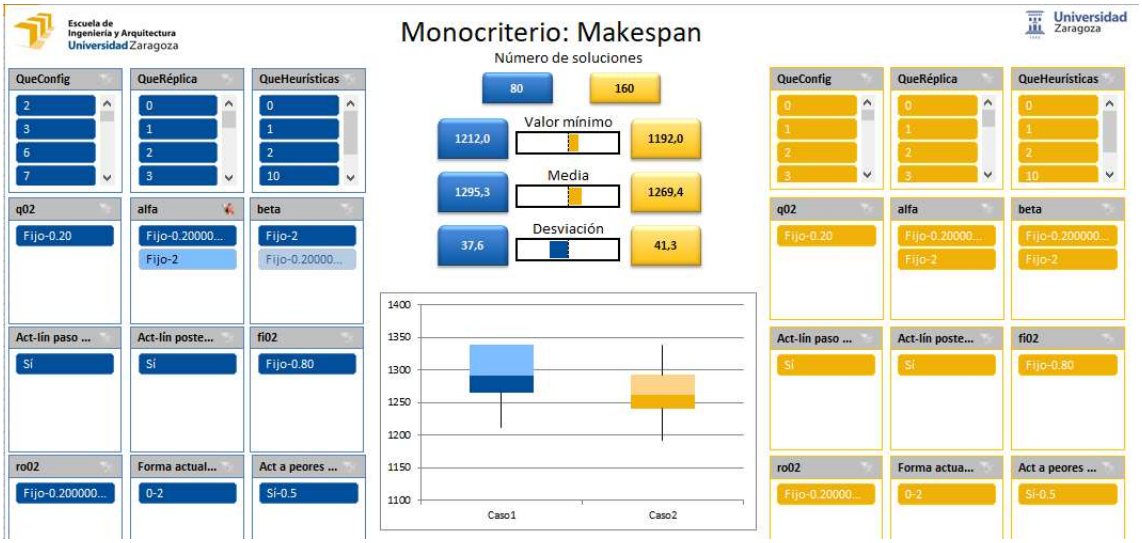



Figura 18: Dashboard para análisis monobjetivo.

## 4.3 Representación de secuencia

Previo al inicio de este trabajo, se contaba con una herramienta para representar la secuencia de los trabajos realizados en cada una de las máquinas a modo de diagrama de Gantt, según se ve en la Figura 19.

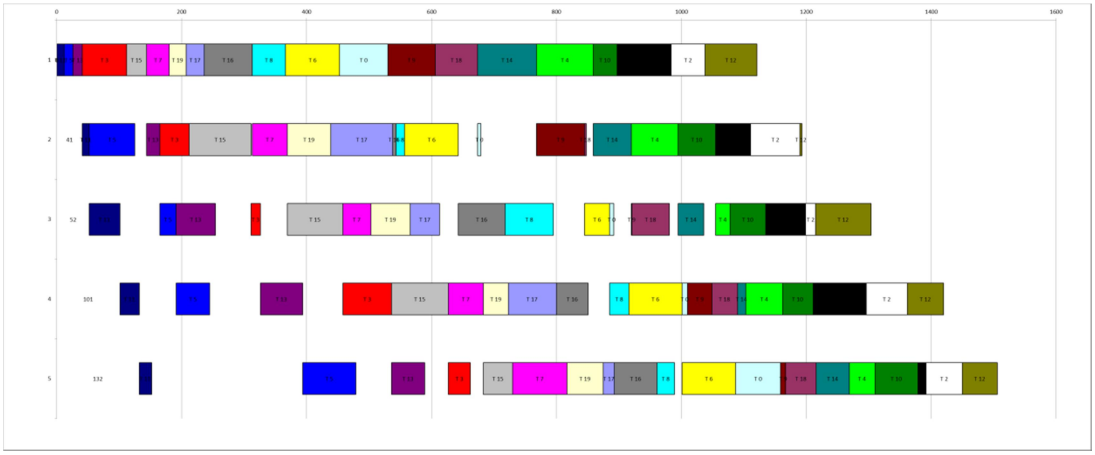


Figura 19: Representación de las secuencias de 20 trabajos en 5 máquinas resuelto en modo permutation.

En este trabajo se ha adaptado esa herramienta para ser funcional junto al dashboard, de tal forma que también sea posible la representación de la secuencia elegida.

# 5. Experimentación

## 5.1 Validación del algoritmo

Dado que se ha modificado el algoritmo MACO para poder resolver los problemas propuestos tanto en modo non-permutation como en modo permutation, resulta necesario comprobar su correcto funcionamiento, para después, realizar comparaciones entre ambos modos.

Como ya se ha comentado, durante (Gutiérrez, 2016) se estudió el algoritmo resolviendo problemas en modo non-permutation y minimizando el makespan, Por ello, en esta primera fase, se va a testear el funcionamiento del algoritmo en modo permutation y con el mismo criterio, y resolviendo los mismos problemas.

Este test de comprobación se ha realizado, en primer lugar, con el problema Taillard 2, de 20 trabajos y 5 máquinas, ya que fue el caso más estudiado y más exitoso en la resolución previa. Además, se conoce su calibración idónea para la resolución en modo non-permutation.

Creamos una batería de experimentaciones combinando todos los parámetros mostrados en la Tabla 1, resultado 88 configuraciones distintas, debiendo realizarse 10 réplicas de 10 segundos para cada una de esas configuraciones. De esta manera, se obtendrán 880 resultados de monobjetivo optimizando el makespan.

| | Alternativas de parámetros | | | | |
|---|---|---|---|---|---|
| Tiempo (s) | 10 | | | | |
| Hormigas | 5 | | | | |
| Heurística | 0 al 10 | | | | |
| $\alpha$ | 0.2 | 2 | | | |
| $\beta$ | 0.2 | 2 | | | |
| $\varphi$ | 0.8 | | | | |
| q0 | 0.2 | 0.8 | | | |
| ML_NumSol | 0 | 1 | | | |

Tabla 1: Configuraciones de parámetros para experimentación en Taillard 2 realizando optimización monobjetivo de makespan.

En este caso, el mejor resultado de makespan recogido en la literatura (Taillard, 1989) con permutation es de 1358. El mejor resultado obtenido en (Gutiérrez, 2016), resolviendo dicho problema en modo non-permutation, fue de 1370. Ahora, tras esta batería de experimentaciones, resolviendo el problema en modo permutation con MACO, el mejor resultado conseguido ha sido de 1371.

Es relevante tener en cuenta que para la mayoría de las configuraciones (54%), se ha obtenido un valor de makespan igual a 1432. En este caso, pese a la elección de distintos parámetros, al parecer el algoritmo tiende a converger al mismo resultado muy fácilmente. Cuando, en contraste, en la resolución de este problema en modo non-permutation no era común que se repitiese la misma solución en más de un 5% del conjunto de las soluciones.

En la Figura 20 se representa la secuencia obtenida en modo permutation (para Taillard 2) donde se observa que ordenar los trabajos en la máquina 1 por orden de tiempos de proceso genera una solución con pocos tiempos ociosos en la última máquina. Teniendo en cuenta que la mejor solución posible, teóricamente, será aquella en donde la última máquina tenga no tenga ningún tiempo ocioso y el inicio del proceso del primer trabajo sea lo antes posible, se considera que es normal que sea complicado mejorar esta solución representada.

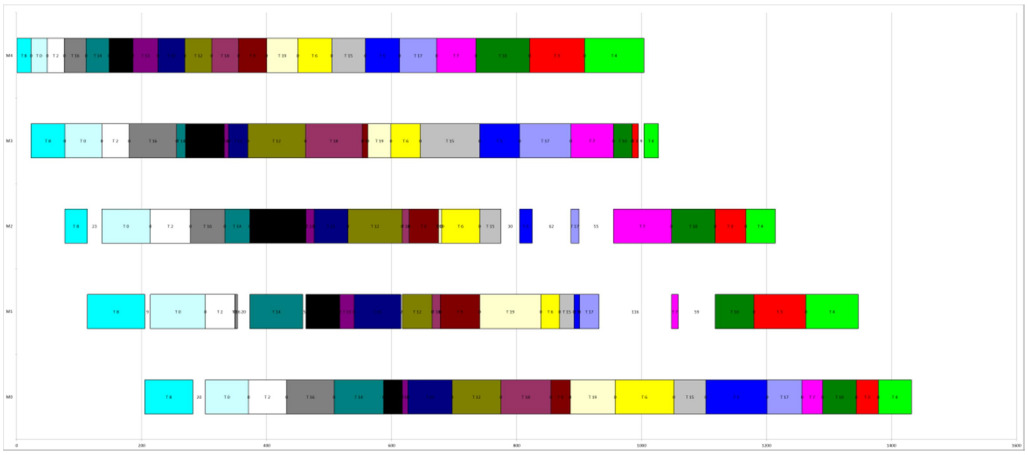


Figura 20: Representación de una solución para la optimización monobjetivo de Taillard 2 mediante el modo permutation.

Utilizando el dashboard, tal y como se puede ver en la Figura 21, se ha podido filtrar y aislar (en el caso azul) las configuraciones con las que se obtienen los mejores resultados. El valor de los parámetros en este caso es: sin mejora local, q0=0.2, α=0.2 y β=2.

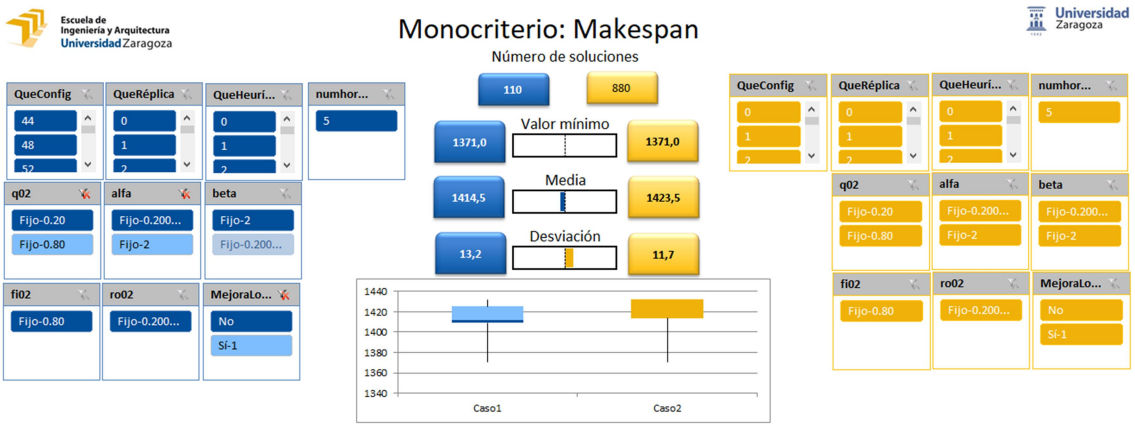



Figura 21: Dashboard tras filtras las mejores configuraciones para resolver el problema de Taillard 2 en modo permutation.

En la Figura 22 se presenta el valor de makespan obtenido en todos los experimentos, en formato de gráfica de barras. En esta se puede ver que: el mejor resultado de makespan es 1371; la media de los 110 resultados es 1414.5; y que el resultado de 1432 (tan repetido en todo el conjunto de experimentos) se encuentra en el cuartil superior, es decir, este resultado se da en menos del 25% de los casos.

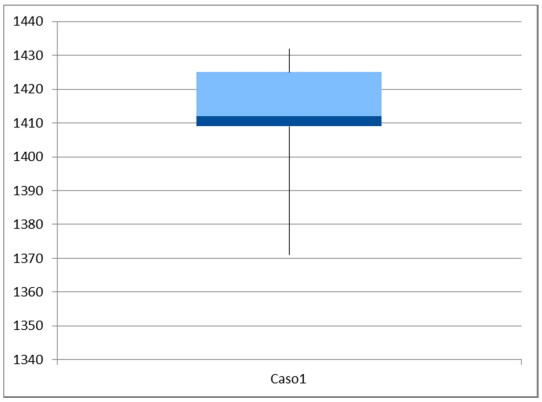



Figura 22: Representación en gráfico de barras del conjunto de soluciones con los parámetros idóneos filtrados para la resolución del problema de Taillard 2 en modo permutation.

Por tanto, se concluye que sigue siendo interesante dar aleatoriedad al algoritmo con un valor de q0=0.2, igual que en el caso de non-permutation. También se mejora no aplicando mejora local y, por último, se ve que merece la pena dar valor a la heurística por encima de la información que da la feromona, aunque este es el parámetro que menos afecta en el conjunto de resultados.

También se ha implementado una gráfica para registrar y visualizar todos los valores de los objetivos, y así ver cómo avanza el algoritmo en la búsqueda de mejores soluciones, y analizar su convergencia, en especial al valor de 1432 que tanto se repite.

En la Figura 23 se muestra los valores de makespan obtenidos a lo largo de una réplica de una configuración del problema resuelto mediante non-permutation. Se puede observar una tendencia a minimizar el makespan a lo largo de las iteraciones, es decir, el algoritmo aprende de sí mismo y va mejorando los resultados. En rojo están marcados los resultados conseguidos tras aplicar una mejora local.

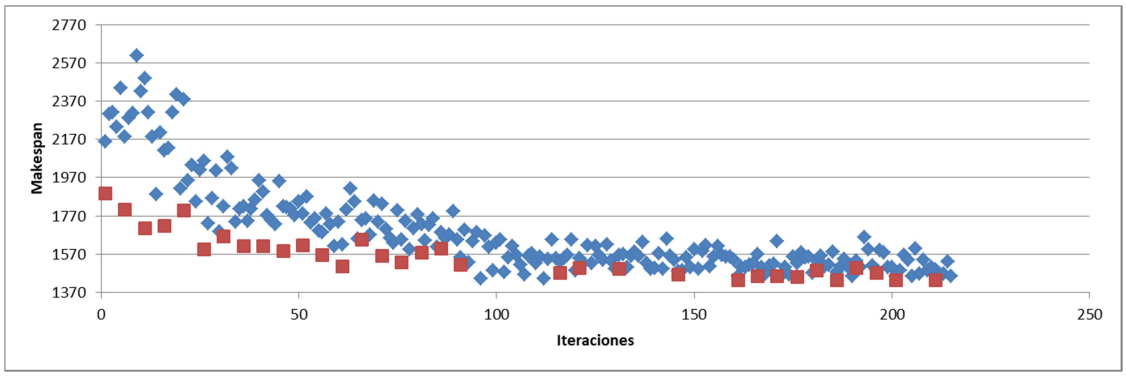



Figura 23: Representación, en el tiempo, de los resultados de makespan obtenidos a lo largo de cada una de las iteraciones, resolviendo el problema en modo non-permutation.

También resulta interesante ver qué resultados se obtienen en cada réplica. En la Figura 24 se recogen los obtenidos en una réplica resolviendo el problema con permutation.

No hay soluciones obtenidas mediante mejora local, pese a que en este caso estaba activada. Esto significa que la mejora aplicada no consigue resultados mejores que los obtenidos mediante el algoritmo básico de la colonia de hormigas.

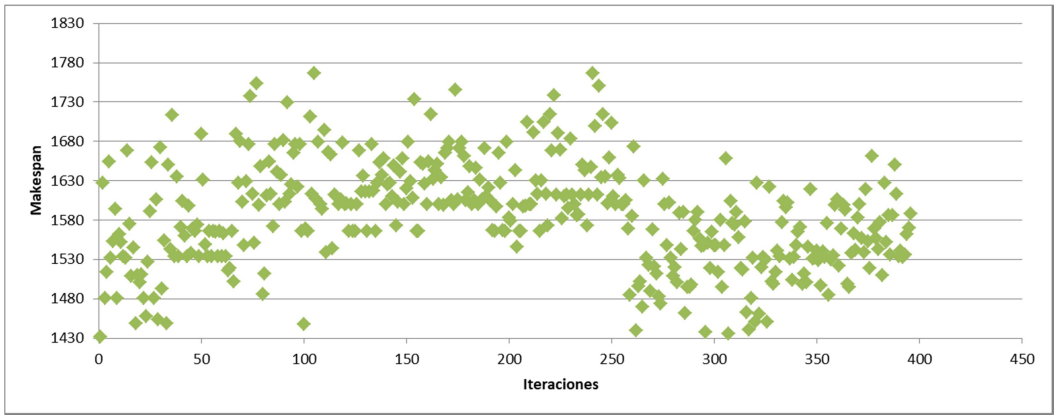



Figura 24: Representación, en el tiempo, de los resultados de makespan obtenidos a lo largo de cada una de las iteraciones, resolviendo el problema en modo permutation.

Analizando el progreso de los resultados en aquellas iteraciones que consiguen 1432 como valor óptimo, se observa que dicha solución se obtiene en la primera iteración, y a lo largo del resto del tiempo de simulación de esa réplica no se consigue mejorar dicho resultado. El criterio que se sigue para la creación de esa primera secuencia de operaciones es ordenarlas de menor a mayor tiempo de proceso en la primera máquina y, al tratarse de una resolución en modo permutation, se repite esa secuencia en el resto de máquinas.

Por último, en la Figura 25, se visualizan juntos todos estos resultados. Al comparar, se aprecia la posibilidad de que si el modo permutation no mejora es porque ha llegado a un límite en su capacidad de optimizar el problema (punto en el que también se encuentra el algoritmo en el modo non-permutation). Por tanto, el algoritmo no funciona necesariamente mal en la optimización de un criterio (makespan) en modo permutation, sino que obtiene rápidamente buenos resultados y le es difícil mejorarlos.

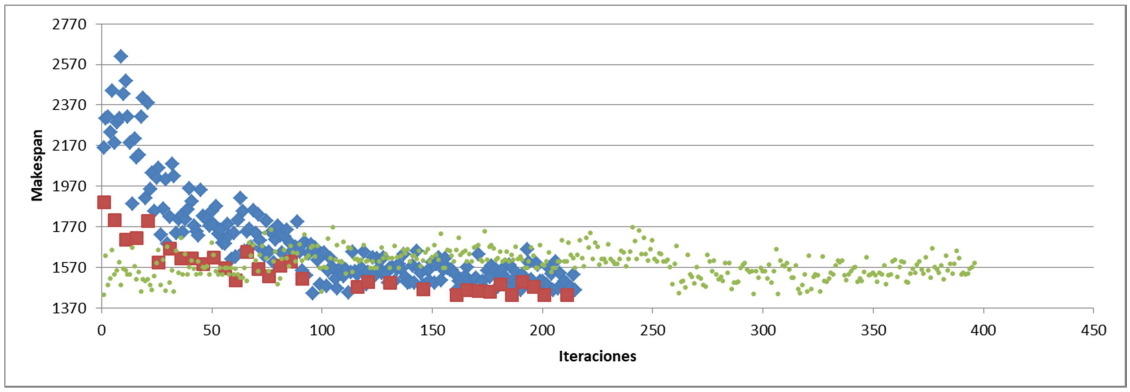



Figura 25: Representación de valores de makespan obtenidos tanto en modo non-permutation (en azul normales y en rojo tras aplicar mejora local) como en modo permutation (verde).

Además, es interesante observar que en la resolución permutation, al realizarse un menor número de decisiones en cada iteración (ya que el algoritmo decide una sola secuencia para todas las máquinas), el número de soluciones obtenidas a lo largo de todo el tiempo de simulación (10 segundos) es mayor (casi 400 soluciones) que en la resolución en modo non-permutation (que no llega a 225 soluciones).

Así, resulta importante intentar proponer una generalización del valor idóneo de los parámetros en la optimización monobjetivo en resolución de problemas en modo permutation. También resulta importante comparar los resultados de los dos modos de configurar este tipo de problemas de secuenciación. Por ello, se han diseñado un conjunto de escenarios, modificando

los parámetros idóneos tras analizar cada conjunto de soluciones, y resolver los otros 9 problemas de Taillard de 20 trabajos y 5 máquinas. Los resultados obtenidos se presentan en la Tabla 2, reflejando si se da o no la circunstancia de que una misma solución se repite en la mayoría de las configuraciones testeadas.

| | Mejor resultado conocido | MACO permutation | ¿Se repite mucho la misma solución? | MACO non-permutation |
|---|---|---|---|---|
| Tai1 | 1278 | 1324 | 1334 se repite el 99% | 1340 |
| Tai2 | 1358 | 1371 | 1432 se repite el 54% | 1370 |
| Tai3 | 1073 | 1149 | No | 1158 |
| Tai4 | 1292 | 1383 | No | 1359 |
| Tai5 | 1231 | 1361 | No | 1380 |
| Tai6 | 1193 | 1273 | 1334 se repite el 60% | 1254 |
| Tai7 | 1234 | 1272 | No | 1263 |
| Tai8 | 1199 | 1284 | 1329 se repite el 80% | 1282 |
| Tai9 | 1210 | 1326 | No | 1290 |
| Tai10 | 1103 | 1192 | No | 1183 |

Tabla 2: Resultados tras la optimización monobjetivo de los problemas de Taillard mediante los modos permutation y non-permutation, así como los mejores resultados conocidos en la literatura para cada problema.

Lo primero que se observa es que solo en 4 problemas de los 10 estudiados, se repiten las mismas soluciones en el conjunto de experimentos. En todos ellos, la solución que se repite se obtiene en la primera iteración.

Por otro lado, comparando los resultados obtenidas en ambos modos, y teniendo en cuenta los mejores resultados obtenidos en la literatura, no se obtiene una conclusión sobre cuál es el mejor modo con el que resolver los problemas, ya que en ciertas ocasiones se obtiene un mejor resultado con permutation y en otros casos con non-permutation.

Por último, con ayuda de la herramienta creada para el análisis de resultados, se han sacado las siguientes conclusiones sobre la parametrización del algoritmo:

- Se cumple, al igual que se daba en el caso de non-permutation, que es mejor que q0 tome valores pequeños para dar aleatoriedad al algoritmo. En la Figura 26, se observan las diferencias de resultados, para el problema Taillard 4, con q0=0.2 y q0=0.8.

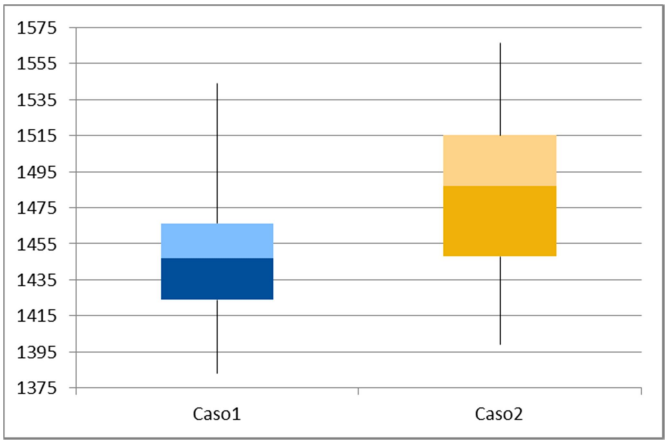



Figura 26: Representación en gráfica de barras de los resultados tras la optimización monobjetivo con q0=0.2 (caso azul) y q0=0.8 (caso naranja).

- Las mejores heurísticas son: 0, 1, 2 y 10.

- En general, es mejor que α tenga un valor más alto que β, ya que así se suele obtener el mejor valor mínimo de makespan, aunque en algunos problemas (tal y como hemos visto antes en el problema de Taillard 2) con α pequeño y β grande se obtiene una mejor media y, además, con una menor desviación.

Tras este conjunto de análisis y comparaciones, ha sido posible evaluar la eficacia del algoritmo en modo permutation para la optimización monobjetivo. Este mismo análisis se realizará en el caso bicriterio.

## 5.2 Elección de banco de problemas

En la literatura hay autores que crean sus propios bancos de problemas y explican el método que han seguido para su generación. Otros acuden a bancos de problemas ya existentes, utilizados por otros autores, y en los que, si es necesario, modifican, o añaden, la información que necesiten para llevar a cabo sus experimentos.

Las instancias más estudiadas y testeadas por los autores que estudian la secuenciación de operaciones son los bancos de problemas de (Dermikol, 1998) y de (Taillard, 1989). En ellos únicamente aparecen los valores de los tiempos de proceso de cada trabajo en cada una de las máquinas. Como uno de los criterios que se va a utilizar es la tardanza, se necesitan bancos de problemas más complejos.

Para la evaluación de las fronteras de Pareto se van a utilizar las métricas propuestas por (Minella, 2014) y, aprovechando que en su trabajo realiza una evaluación de 23 algoritmos con un banco de problemas conocido, se ha decidido utilizar las mismas instancias que utiliza. Éstas tienen los tiempos de proceso de (Taillard, 1989) y añade los las fechas de entrega de los trabajos siguiendo el enfoque dado en (Hasija, 2004), que consiste en calcular la fecha de entrega $d_j$, como $d_j = P_j x(1 + random * 3)$ siendo $P_j$ la suma de todos los tiempos de proceso del trabajo j en cada máquina, y siendo $random$ un número aleatorio uniformemente distribuido entre [0,1].

Dichas problemas pueden ser descargados en http://soa.iti.es/instancias-de-problemas. En este trabajo, utilizaremos los 10 problemas de 20 trabajos y 5 máquinas.

## 6.3 Esquema de experimentación

Una vez definidos los problemas, presentado el algoritmo a utilizar, y elegidas las métricas con las que se evaluarán las soluciones, se plantea el procedimiento a seguir para la calibración del algoritmo y, a su vez, obtención de resultados.

Debido a que hay 15 parámetros de los que depende el algoritmo resulta inviable realizar la experimentación completa combinando todo los valores de todos ellos. Por tanto, se va a realizar una calibración por grupos de parámetros.

El problema, para cada configuración de parámetros, se resolverá en modo permutation y non-permutation. Así, primero, se podrá comprobar si la parametrización del algoritmo es la misma para ambos casos, y, segundo, se podrá ver cuál de los dos modos es mejor para realizar optimizaciones multicriterio.

Para el testeo y calibración del algoritmo se van a escoger aquellos problemas de Taillard (de 20 trabajos y 5 máquinas y con las fechas de entrega de (Minella, 2014) en los que en la optimización monobjetivo no ha habido una repetición por encima del 50% de un mismo resultado. En estos casos es posible que sea más difícil concretar con qué parámetros se obtienen mejores resultados por repetirse siempre la misma solución.

Se comenzará por el problema Taillard 3, se hará una experimentación de un gran número de configuraciones de la que ya se obtendrán algunas conclusiones, y con esa información se procederá a realizar una batería de configuraciones distinta con el problema de Taillard 4. Así, problema tras problema, se irán eligiendo los parámetros que sea evidente su calibración y se irán dejando abiertos aquellos en los que no se obtengan conclusiones claras.

Por último, tras realizar una calibración a lo largo de todos los problemas, se volverá a lanzar una última resolución, con una pequeña batería de configuraciones común, a todos los problemas, y así, y comparando con los resultados antes conseguidos y con los de otros autores, se podrá valorar si la parametrización del algoritmo ha sido correcta.

Toda esta experimentación lleva mucho tiempo de computación tanto por la propia optimización de los problemas como luego en su evaluación. El cálculo del hipervolumen y épsilon con el dashboard supone un tiempo que hay que tener en cuenta. Por ello, en las primeras calibraciones comentadas se configurará el archivo Excel para que el de épsilon no sea tan concreto y tan preciso como el que usaremos para evaluar finalmente el algoritmo con las configuraciones finales.

## 5.4 Calibración del algoritmo

Se comienza optimizando el problema de Taillard 3 realizando una batería de simulaciones con la combinación de configuraciones de la Tabla 3:

| | Alternativas de parámetros | | | | |
|---|---|---|---|---|---|
| Tiempo (s) | 10 | | | | |
| Hormigas | 5 | | | | |
| Heurística | 0 al 10 | | | | |
| $\alpha$ | 0.2 | 1 | 2 | | |
| $\beta$ | 2 | 1 | 0.2 | | |
| $\varphi$ | 0.2 | 0.5 | 0.8 | | |
| q0 | 0.2 | 0.5 | 0.8 | | |
| Act. a poster | 0 | | | | |
| $\rho$ | 0 | | | | |
| ML_NumSol | 0 | | | | |

Tabla 3: Combinación de parámetros para la experimentación en el problema de Taillard 3 realizando una optimización multiobjetivo de makespan y tardiness.

La combinación de todos estos parámetros supone 297 configuraciones distintas y de cada una de ellas se realizarán 10 réplicas. El total de experimentos se duplica al resolver el problema de los dos modos, permutation y non-permutation. El tiempo de computación total necesario es de 16 horas y media.

La primera comparación es la diferencia entre el conjunto de resultados resueltos en modo permutation (caso azul) y en modo non-permutation (caso naranja), tal y como se ve en la Figura 27. Los puntos morados en la gráfica, definidos como “Mejor Pareto Conocido”, corresponden a los mejores valores obtenidos en el trabajo de (Minella, 2014):

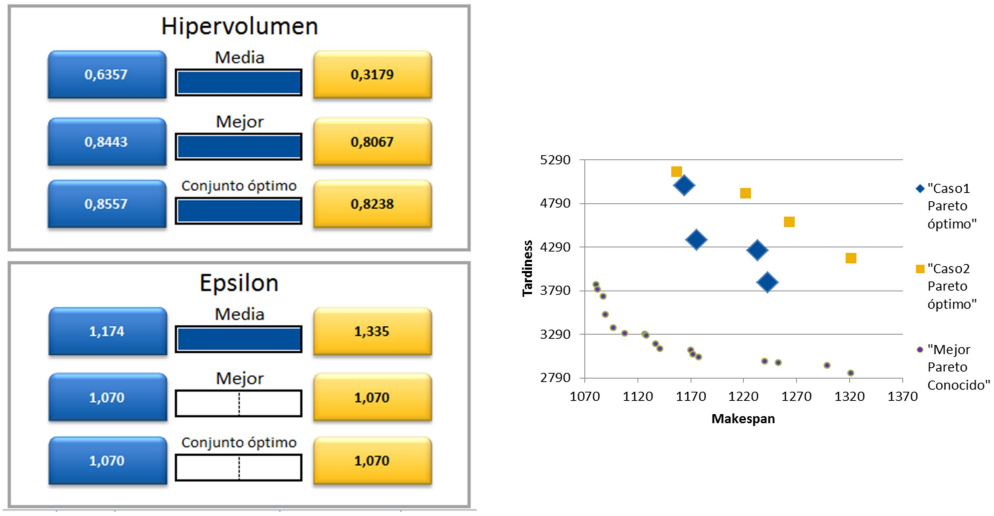



Figura 27: Comparación entre modo permutation (caso azul) y modo non-permutation (caso naranja) en la optimización del problema de Taillard 3.

Tanto por el resultado de las métricas como por la representación de los mejores valores de cada conjunto en la gráfica, se ve que la resolución en modo permutation da muchos mejores resultados que en modo non-permutation.

El primer parámetro dentro de la calibración del algoritmo es la definición de q0. Si se filtra agrupando por un lado todas las configuraciones realizadas con q0=0.2 (caso azul) y por otro lado con todas las realizadas con q0=0.5 y 0.8 (caso naranja), se obtienen los resultados de la Figura 28.

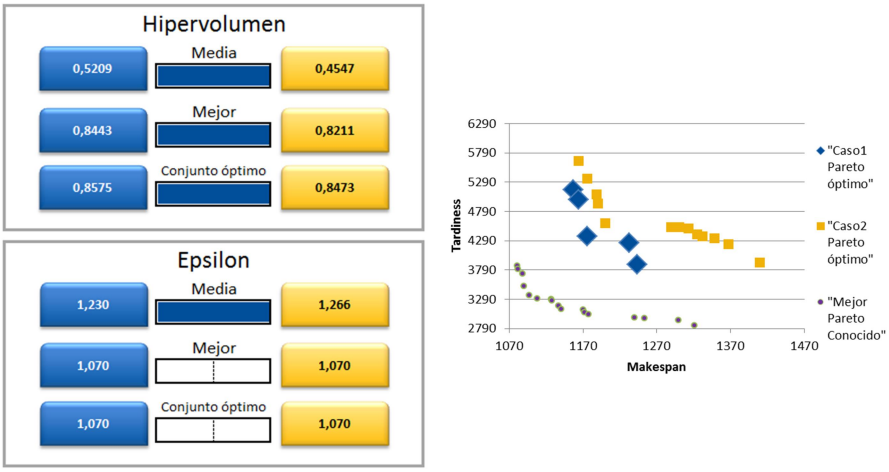



Figura 28: Comparación entre los conjuntos con q0=0.2 (caso azul) y q0=0.5 y 0.8 (caso naranja) en la optimización del problema de Taillard 3.

Se aprecia cómo se mejoran los resultados si q0=0,2. Esta conclusión es independiente de si realizamos esta comparación filtrando por el modo de resolución del problema.

Filtrando aquellas soluciones obtenidas en los casos en que q0=0.2, el siguiente paso es comparar las configuraciones en función de los valores de $\alpha$ y $\beta$. En la Figura 29 se representa en el caso azul los valores con ($\alpha$=2; $\beta$=0.2) y ($\alpha$=1; $\beta$=1) y, en el caso naranja, se representa ($\alpha$=0.2; $\beta$=2)., dando mejores resultados el caso azul.

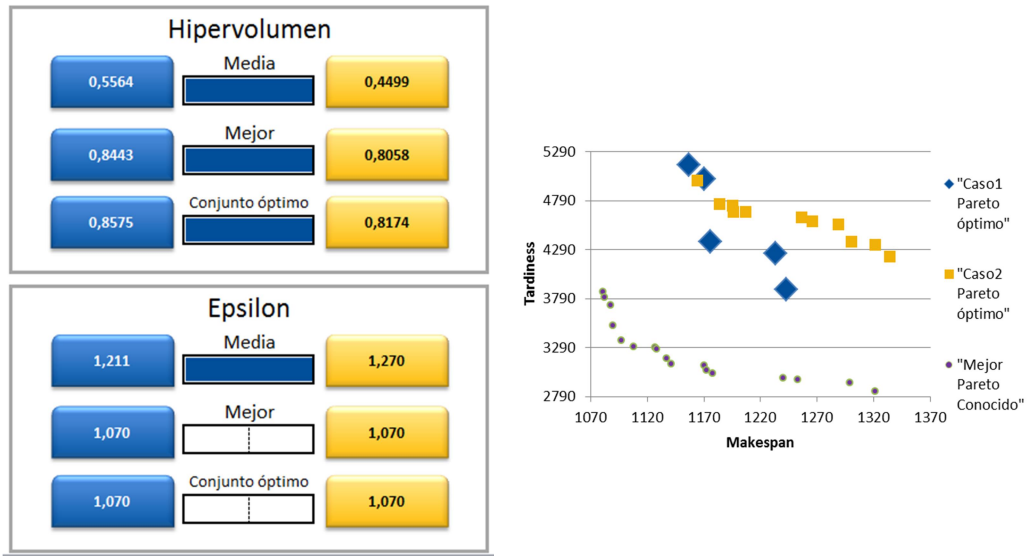



Figura 29: Con el filtro de q0=0.2 comparación entre el conjunto de soluciones con $\alpha$=2 $\beta$=0.2 y $\alpha$=1 $\beta$=1 (caso azul) con $\alpha$=0.2 $\beta$=2 (caso naranja) en la optimización del problema de Taillard 3.

Además se ha realizado esta comparación filtrando para cada una de las heurístitcas, en ambos modos, y buscando si en alguna situación un valor alto de $\beta$ (y, por tanto, darle peso a la información que aporta la heurística) conviene en la resolución del problema. Sin embargo, en todos los casos se llega a la conclusión de que son mejores las configuraciones ($\alpha$=2; $\beta$=0.2) y ($\alpha$=1; $\beta$=1).

En la Figura 30 se comparan las 2 configuraciones anteriores, ($\alpha$=2; $\beta$=0.2) en el caso azul y ($\alpha$=1; $\beta$=1) en el caso naranja para ver cuál es ellas ofrece mejores resultados.

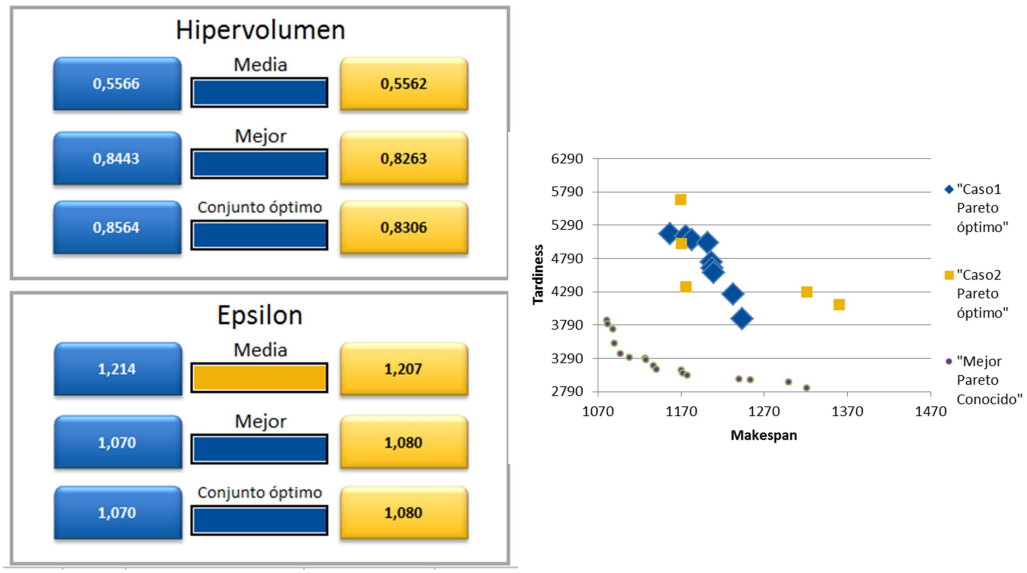



Figura 30: Comparación de conjuntos de valores con $\alpha$=2 $\beta$=0.2 (caso azul) y $\alpha$=1 $\beta$=1 (caso naranja) en la optimización del problema de Taillard 3.

Se ve que los valores de las métricas están muy parejos, y ello se demuestra también en la gráfica, con dos frentes de Pareto distintos y en el que ninguno domina al otro. En este caso no se concluye nada concreto, por lo que en siguientes experimentos se estudiará si los resultados obtenidos son mejores con valor de $\alpha$ bajo y $\beta$ alto o al contrario.

Por último, manteniendo los filtros de q0=0,2, $\alpha$=2 y $\beta$=0.2, se representa en el lado azul de la Figura 31 los resultados con las heurísticas 1, 2, 3 y 10, que previamente se ha filtrado individualmente y reconocido como las mejores, y en el naranja, el resto de heurísticas. Se observa que el frente de Pareto del primer caso es mejor que el segundo.

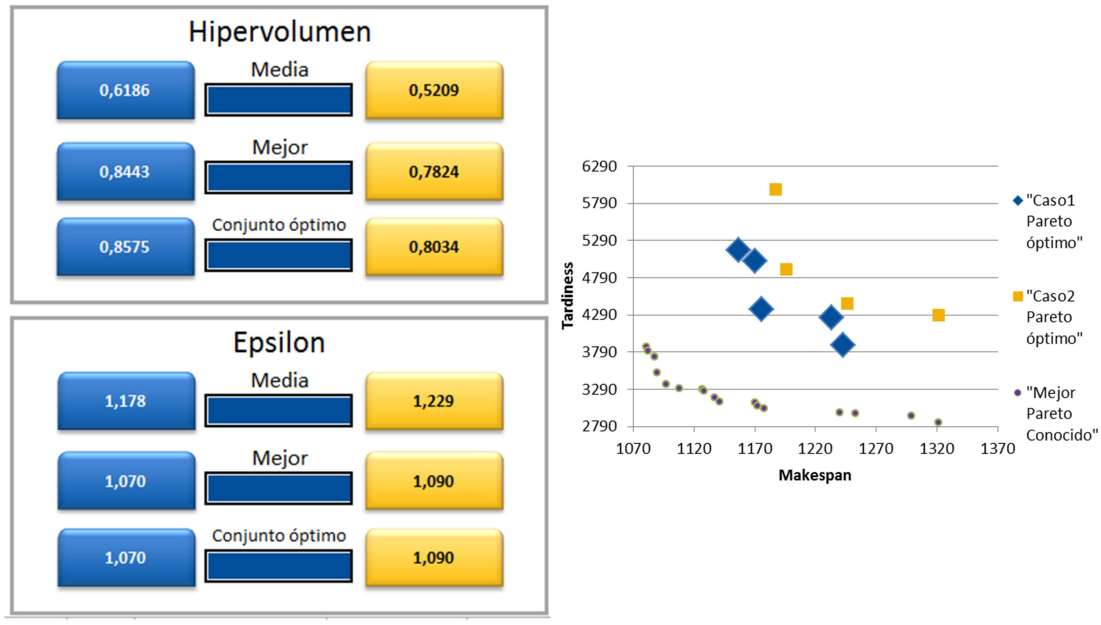



Figura 31: Comparación entre conjuntos de soluciones. Heurísticas 1, 2, 3 y 10 (caso azul) y heurísticas 0, 4, 5, 6, 7, 8 y 9 (caso naranja).

En cambio, comparando entre valores de $\varphi$ no obtenemos ninguna conclusión clara, así que continuaremos usando distintos valores de ese parámetro para ver si en futuros problemas podemos decidir cuál es más óptimo.

A partir de estas conclusiones, se generan la siguiente batería de escenarios (ver Tabla 4), más pequeña, y que probaremos en los problemas de Taillard 4 y 5.

| | Alternativas de parámetros | | | | |
|---|---|---|---|---|---|
| Tiempo (s) | 10 | | | | |
| Hormigas | 5 | 10 | | | |
| Heurística | 0 | 1 | 2 | 10 | |
| $\alpha$ | 0.2 | 2 | | | |
| $\beta$ | 2 | 0.2 | | | |
| $\varphi$ | 0.2 | 0.8 | | | |
| q0 | 0.2 | | | | |
| Act. a poster | 0 | 1_1 | | | |
| $\rho$ | 0.2 | | | | |
| ML_NumSol | 0 | | | | |

Tabla 4: Combinación de parámetros para la experimentación los problemas de Taillard 4 y 5 realizando una optimización multiobjetivo de makespan y tardiness.

Si se combinan todos los valores de los parámetros entre sí, hay 68 configuraciones distintas, lo que supone un total de 3.5 horas de experimentación con cada uno de los problemas.

Con los resultados obtenidos, se refutan algunas de las conclusiones anteriores como, por ejemplo, que, de nuevo, el modo permutation obtiene mejores fronteras de Pareto que el non-permutation.

Además se vuelve a apreciar que se obtienen mejores frentes de Pareto con $\alpha$=2 y $\beta$=0.2 que con $\alpha$=0.2 y $\beta$=2, por lo que en las siguientes baterías no habrá configuraciones con un valor bajo de $\alpha$ y alto de $\beta$.

En cuanto al valor de $\varphi$, aunque el valor de las métricas en función de los dos parámetros usados muestra diferencias muy sutiles, se mantienen las mismas conclusiones tanto en problema de Taillard 3, como en el 4 y en el 5. De esta manera, es mejor usar un valor de $\varphi$, tal y como se aprecia en la Figura 32 donde se representa $\varphi$=0.8 en el caso azul y $\varphi$=0.2 en el caso naranja.

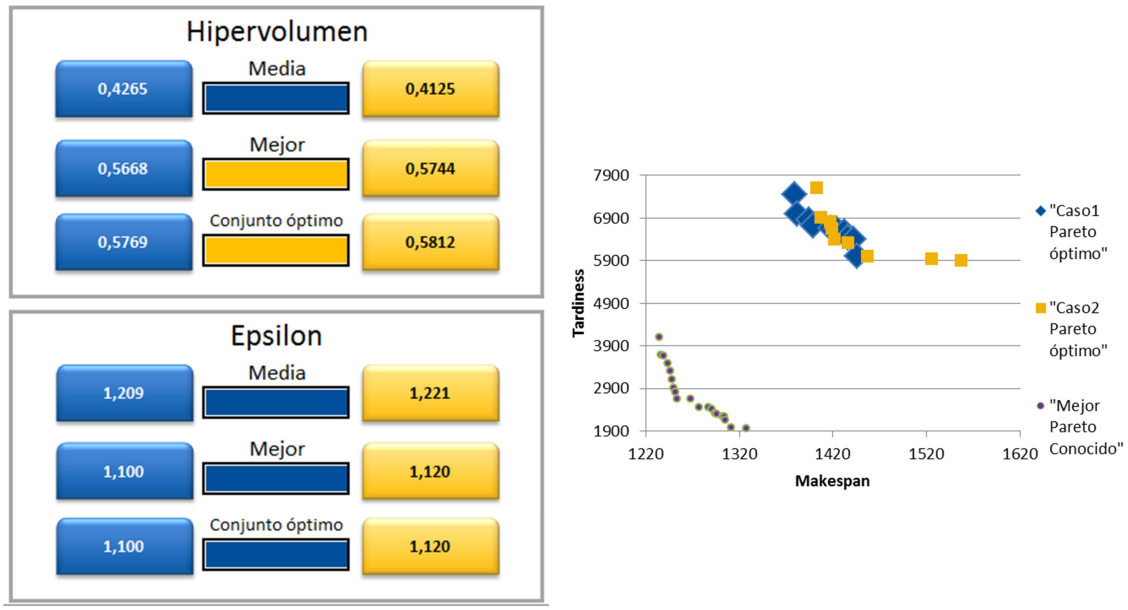



Figura 32: Comparación entre valores de $\varphi$=0.8 (caso azul) y $\varphi$=0.2 (caso naranja) en la optimización del problema de Taillard 5.

Los indicadores de calidad son muy parejos, y en la gráfica no se aprecia ninguna dominación fuerte de ningún frente de Pareto. Sin embargo, como hemos comentado, aunque la diferencia es pequeña siempre se concluye que es mejor un valor de $\varphi$=0.8 por lo que será ese el valor del parámetro con el que seguiremos realizando las siguientes pruebas.

Para finalizar esta primera calibración, se estudia la influencia del número de hormigas utilizadas. En el modo permutation, aunque los resultados son ligeramente mejores cuando hay 10 hormigas, no hay grandes diferencias entre las fronteras de Pareto de ambos conjuntos. Sin embargo, en el modo de non-permutation en el problema de Taillard 5 los resultados para 5 y 10 hormigas se representan en la Figura 33 en el lado azul y naranja respectivamente.

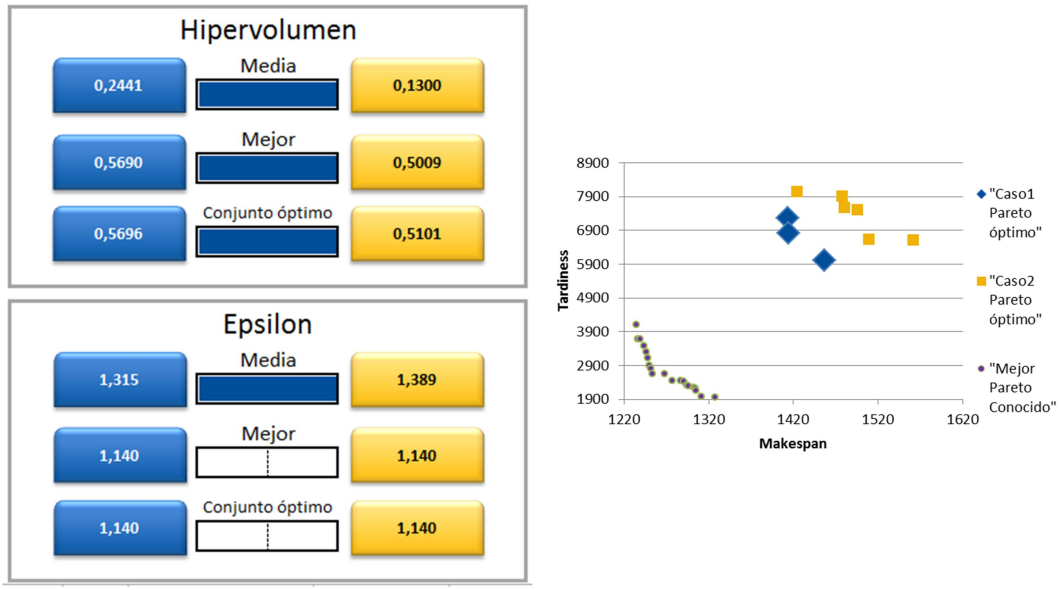



Figura 33: Representación, tras filtrar en modo non-permutation, el conjunto de soluciones realizados con 5 hormigas (caso azul) y el realizado con 10 hormigas (caso naranja) en la optimización del problema de Taillard 5.

Tanto en los indicadores de calidad como en la representación en la gráfica se muestra que el uso de 5 hormigas es claramente mejor que el de 10 en los problemas resueltos en modo non-permutation. Esta diferencia en los frentes de Pareto se aprecia igual en el problema de Taillard 4.

Otros dos parámetros críticos a analizar en esta batería de experimentos es la actualización a posteriori. Por tanto, en el siguiente diseño de experimentaciones, que testearemos en el problema de Taillard 7, incidimos en valorar la actualización a posteriori sobre varias soluciones y probaremos realizar simulaciones con más hormigas para comprobar si es beneficioso para la optimización en modo permutation.

| | Alternativas de parámetros | | | | |
|---|---|---|---|---|---|
| Tiempo (s) | 10 | | | | |
| Hormigas | 5 | 20 | | | |
| Heurística | 0 | 1 | 2 | 10 | |
| α | 2 | | | | |
| β | 0.2 | | | | |
| φ | 0.8 | | | | |
| q0 | 0.2 | | | | |
| Act. a poster | 0 | 1_1 | 1_3 | | |
| P | 0.2 | 0.8 | | | |
| ML_NumSol | 0 | | | | |

Tabla 5: Combinación de parámetros para la experimentación en el problema de Taillard 7 realizando una optimización multiobjetivo de makespan y tardiness.

En este caso, la combinación de estos parámetros significa 40 configuraciones distintas, lo que supone 2 horas de experimentación.

Tras la experimentación, la primera conclusión es que con más de 5 hormigas en el modo non-permutation, las fronteras de Pareto obtenidas son peores. En cambio, en el modo permutation se observa que es ligeramente mejor usar 20 hormigas (caso azul) que 5 hormigas (caso naranja) según Figura 34.

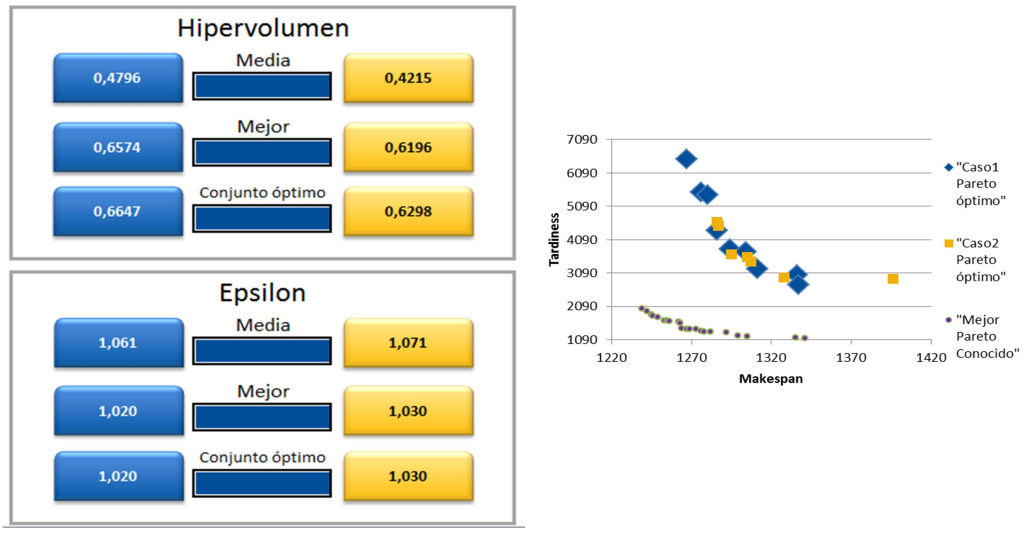



Figura 34: Comparativa, en modo de resolución permutation, entre el conjunto de soluciones obtenido con 20 hormigas (caso azul) y con 5 hormigas (caso naranja) en la optimización del problema de Taillard 7.

Con las pruebas del siguiente problema se comprobará si merece la pena aumentar aún más el número de hormigas.

En cuanto al tipo de actualización a posteriori de nuevo las conclusiones no son claras. Parece que en el caso de permutation es mejor actualizar con 1 ó 3 soluciones, en vez de no hacerlo. En el caso de non-permutation es mejor no hacerlo. Sin embargo, se continuará intentando detallar más esta calibración en siguientes experimentos.

Por otro lado, el análisis de las heurísticas va cambiando a lo largo de los distintos problemas: en ocasiones el uso de la heurística 10 es claramente mejor para casos non-permutation, y en otros no se da esa circunstancia. Sin embargo, sí se ve que la heurística 0 no está siendo buena ni en este problema ni en los anteriores, por tanto, en los siguientes tests se comprobarán las heurísticas 1, 2, 3 y 10.

La combinación de parámetros a testearemos en el problema de Taillard 9 es la siguiente:

| | Alternativas de parámetros | | | | |
|---|---|---|---|---|---|
| Tiempo (s) | 10 | | | | |
| Hormigas | 5 | 20 | 40 | | |
| Heurística | 1 | 2 | 3 | 10 | |
| $\alpha$ | 2 | | | | |
| $\beta$ | 0.2 | | | | |
| $\varphi$ | 0.8 | | | | |
| q0 | 0.2 | | | | |
| Act. a poster | 0 | 1_3 | 1_5 | | |
| P | 0.2 | | | | |
| ML_NumSol | 0 | | | | |

Tabla 6: Combinación de parámetros para la experimentación en el problema de Taillard 9 realizando una optimización multiobjetivo de makespan y tardiness.

Con esta batería de experimentos hay 36 configuraciones distintas para cada modo de resolución, lo cual supone 2 horas de computación.

En primer lugar, se compara cómo afecta el número de hormigas en el modo permutation, y se ve que sigue siendo mejor tener un número de hormigas mayor que 5, pero no hay diferencia entre usar 20 hormigas y 40. Por ello, se piensa que 20 hormigas en modo permutation es un

tope en el que no se mejoran los resultados, concluimos que con se pueden obtener los mejores resultados.

Respecto a la actualización o no a posteriori de la feromona, en la Figura 35 se comparan los resultados si no se actualiza (caso azul) y, si se actualiza (caso naranja), en el modo de resolución non-permutation.

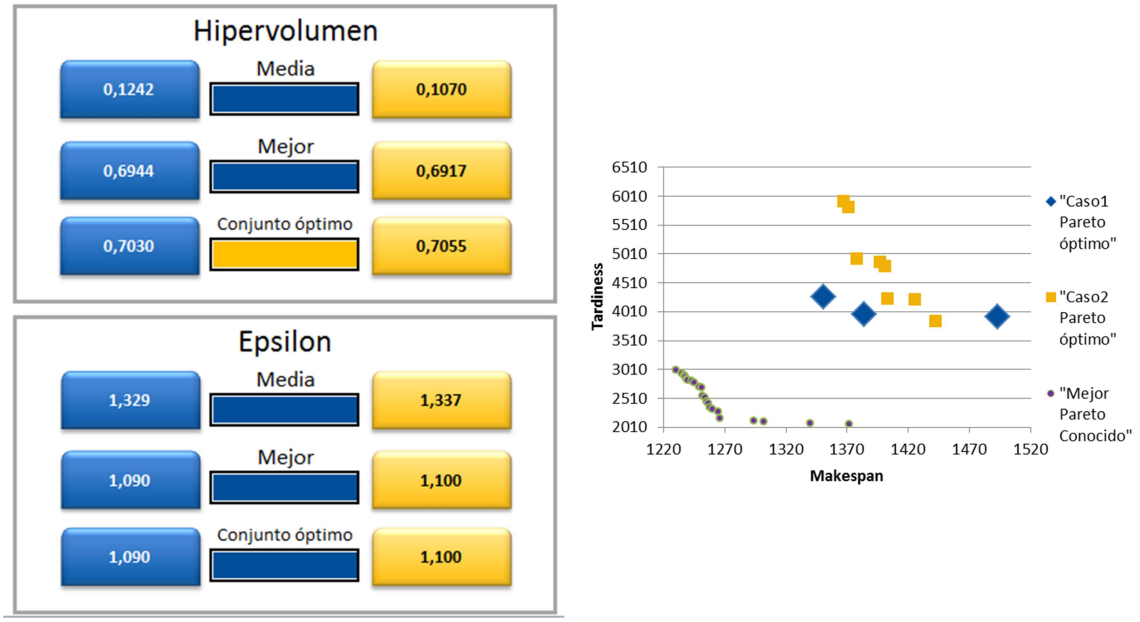



Figura 35: Comparativa entre el conjunto de soluciones donde no se actualiza a posterior (azul) y el conjunto de soluciones en el que sí que se actualiza a posteriori (naranja) en la optimización del problema de Taillard 9.

Se observa que, aunque el frente de Pareto azul contiene muchas menos soluciones prácticamente domina al frente de Pareto naranja. Además, salvo el indicador de hipervolumen para el conjunto formado por todas las soluciones, el resto de resultados reflejan que es mejor no hacer actualización a posteriori en el modo non-permutation.

En la siguiente experimentación, realizada con el problema de Taillard 10, se aborda, de forma más extensa, la actualización a posteriori, para comprobar cuánto condiciona la solución en el modo permutation. También se introduce la posibilidad de aplicar mejora local (característica que no añadida anteriormente ya que en el análisis monobjetivo no se observaba que fuese conveniente aplicar dicha mejora).

| | Alternativas de parámetros | | | | |
|---|---|---|---|---|---|
| Tiempo (s) | 10 | | | | |
| Hormigas | 5 | 20 | | | |
| Heurística | 1 | 2 | 3 | 10 | |
| $\alpha$ | 2 | | | | |
| $\beta$ | 0.2 | | | | |
| $\varphi$ | 0.8 | | | | |
| q0 | 0.2 | | | | |
| Act. a poster | 0 | 1_5 | | | |
| $\rho$ | 0.2 | 0.5 | 0.8 | | |
| ML_NumSol | 0 | 1_1 | 1_3 | 1_5 | |

Tabla 7: Combinación de parámetros para la experimentación en el problema de Taillard 10 realizando una optimización multiobjetivo de makespan y tardiness.

En este último análisis se ha profundizado más entre las combinaciones de configuraciones buscando la parametrización ideal. En el caso de permutation, la mejor opción ha sido utilizar 20 hormigas, aplicando una mejora local a 3 de ellas y realizando una actualización a posteriori a 5 soluciones con un factor de $\rho$=0.5 (y en este caso el uso de la heurística 10 nunca es una buena opción).

En cambio, en el caso de non-permutation la configuración idónea se ha conseguido con 5 hormigas, sin actualización a posteriori, y tanto aplicando una mejora local de 3 como de 5 soluciones.

Con las conclusiones obtenidas a lo largo de las experimentaciones realizadas, se han diseñado dos baterías de configuraciones, una para resolución permutation y otra para resolución non-permutation. Con ellas, se resolverán todos los problemas de Taillard y se compararán los indicadores de calidad con los resultados anteriormente obtenidos como con los de otros autores.

Las configuraciones "optimas" aparecen en la Tabla 8:

| | Permutation | Non-permutation |
|---|---|---|
| Tiempo (s) | 10 | 10 |
| Hormigas | 20 | 5 |
| Heurística | 1, 2 ,3 | 1, 2, 3, 10 |
| $\alpha$ | 2 | 2 |
| $\beta$ | 0.2 | 0.2 |
| $\varphi$ | 0.8 | 0.8 |
| q0 | 0.2 | 0.2 |
| Act. a poster | 1_5 | 0 |
| $\rho$ | 0.5 | 0.5 |
| ML_NumSol | 1_3 | 1_3, 1_5 |

Tabla 8: Configuraciones óptimas tras la calibración del algoritmo en la optimización multiobjetivo de los problemas de Taillard. Subrayados aquellos parámetros que difiere su calibración en función del modo de resolución del problema.

## 5.5 Análisis de resultados

A continuación, y, en primer lugar, se va a comprobar la eficacia de la calibración del algoritmo. Para ello se compararán los resultados obtenidos durante la experimentación con los obtenidos cuando los parámetros toman el valor "ideal", según Tabla 8.

En el dashboard utilizado para analizar los frentes de Pareto se calcula el hipervolumen y épsilon. También se calculan la media del indicador de calidad de todos los conjuntos de soluciones, el indicador de calidad del mejor conjunto dentro de todas las soluciones y el indicador de calidad del supuesto frente de Pareto que se generaría con todas las simulaciones de todas las configuraciones realizadas.

Teniendo en cuenta que la calibración se ha hecho con un número de configuraciones mucho mayor que los experimentos finales resulta injusto comparar el segundo y tercer valor comentado. Por ello, en la Figura 36 se refleja exclusivamente la media de los valores de hipervolumen y épsilon para cada batería de experimentos realizada en cada uno de los problemas estudiados. La columna C recoge los valores de conseguidos con los experimentos de calibración; F con los parámetros que se ha decidido que eran los mejores.

| | Hipervolumen | | | |
|---|---|---|---|---|
| | Permutation | | Non-Permutation | |
| | C | F | C | F |
| Taillard 1 | | 0,6192 | | 0,224 |
| Taillard 2 | | 0,6693 | | 0,4489 |
| Taillard 3 | 0,6357 | 0,7548 | 0,3179 | 0,5447 |
| Taillard 4 | 0,7673 | 0,7943 | 0,4913 | 0,6371 |
| Taillard 5 | 0,4882 | 0,5108 | 0,187 | 0,3577 |
| Taillard 6 | | 0,7366 | | 0,5508 |
| Taillard 7 | 0,4506 | 0,5082 | 0,0102 | 0,2401 |
| Taillard 8 | | 0,7861 | | 0,5778 |
| Taillard 9 | 0,6636 | 0,6766 | 0,1133 | 0,4958 |
| Taillard 10 | 0,697 | 0,737 | 0,4323 | 0,5032 |
| Media | 0,6171 | 0,6793 | 0,2587 | 0,4580 |

| | Epsilon | | | |
|---|---|---|---|---|
| | Permutation | | Non-Permutation | |
| | C | F | C | F |
| Taillard 1 | | 1,04 | | 1,134 |
| Taillard 2 | | 1,03 | | 1,093 |
| Taillard 3 | 1,174 | 1,123 | 1,335 | 1,224 |
| Taillard 4 | 1,107 | 1,099 | 1,283 | 1,196 |
| Taillard 5 | 1,162 | 1,158 | 1,352 | 1,254 |
| Taillard 6 | | 1,096 | | 1,189 |
| Taillard 7 | 1,066 | 1,066 | 1,177 | 1,118 |
| Taillard 8 | | 1,098 | | 1,205 |
| Taillard 9 | 1,105 | 1,103 | 1,333 | 1,181 |
| Taillard 10 | 1,12 | 1,115 | 1,251 | 1,203 |
| Media | 1,1223 | 1,0928 | 1,2885 | 1,1797 |

Figura 36: Resultados de hipervolumen y épsilon de los 10 problemas de Taillard durante la calibración (columna C) y con los parámetros "ideales" (columna F), marcando en verde, frente al rojo, aquellos problemas en los que se ha mejorado el resultado.

Comparando los resultados, se observa que ha habido una mejora en los resultados medios de los indicadores de calidad de todos los problemas en los que antes habíamos realizado optimizaciones con configuraciones que no necesariamente eran las mejores.

Por otro lado, se confirma que siempre es mejor resolver el problema en modo permutation, aunque haya menos combinaciones posibles que permitan resolver el problema. Sin embargo, por otro lado, es importante tener en cuenta que la calibración resulta más efectista en modo non-permutation.

Es interesante el hecho de que la optimización mediante el modo permutation sea, de forma tan evidente, mejor que la optimización non-permutation cuando, como explicábamos en la descripción del problema de secuenciación, dentro de las soluciones posibles en el caso de non-permutation se encuentran las de permutation. Esto significa que, en el modo permutation, al tener menos posibilidades entre las que buscar, optimiza mejor buscando óptimos locales cercanos a las soluciones que ya ha encontrado, frente a la resolución non-permutation, que al tener tantas combinaciones posibles de secuencias, no consigue encontrar una optimización tan buena para los problemas.

Tras la comparación realizada entre los resultados obtenidos por el algoritmo sin calibrar con los obtenidos con los parámetros calibrados queda, por último, comparar con otros algoritmos que hayan estudiado esos mismos problemas en las mismas condiciones.

Ya que hemos estado siguiendo, tanto sus métricas como sus bancos de problemas, acudimos a los resultados aportados por (Minella, 2014). En su trabajo se comparan 23 algoritmos distintos resolviendo los problemas de Taillard en modo permutation. La información aportada es la media de los indicadores de épsilon tras optimizar las 110 instancias de Taillard (en el trabajo presente se ha basado en las 10 primeras, que son las más pequeñas), realizando la experimentación un tiempo proporcional al tamaño del problema (de forma que (Minella, 2014), al igual que en este trabajo, dedica 10 segundos a los problemas con los que hemos realizado la experimentación).

En la

Figura 37 se presenta el valor de épsilon del estudio Minella(2014), junto con los resultados de MACO calibrado funcionando en modo permutation como en modo non-permutation. Es importante recordar que los resultados serán mejores en cuanto más cercanos estén a 1.

| Posición | | Épsilon |
|---|---|---|
| 1 | MOSAII | 1,056 |
| 2 | MOGALS | 1,08 |
| 3 | PESA | 1,081 |
| 4 | PESAII | 1,084 |
| 5 | MACO permut | 1,093 |
| 6 | PGA_ALS | 1,118 |
| 7 | MACO permut | 1,122 |
| 8 | MOTS | 1,129 |
| 9 | MOGA_Murata | 1,138 |
| 10 | CMOGA | 1,143 |
| 11 | CNSGAII | 1,154 |
| 12 | NSGAII | 1,155 |
| 13 | SPEA | 1,155 |

| Posición | | Épsilon |
|---|---|---|
| 14 | MACO non-permut | 1,180 |
| 15 | ε-NSGAII | 1,182 |
| 16 | (μ+λ)-PAES | 1,183 |
| 17 | ε-MOEA | 1,216 |
| 18 | PAES | 1,23 |
| 19 | MACO non-permut | 1,289 |
| 20 | MOSA_Suresh | 1,303 |
| 21 | PILS | 1,363 |
| 22 | SA_Chakravarty | 1,41 |
| 23 | A-IBEA | 1,514 |
| 24 | ENGA | 1,542 |
| 25 | SPEAII | 1,549 |
| 26 | NSGA | 1,593 |
| 27 | B-IBEA | 1,637 |

Figura 37: Comparativa de resultados entre los aportados por (Minella, 2014) y los conseguidos por el algoritmo MACO, en rojo resultados durante la calibración y en verde con los parámetros considerados como ideales.

Observamos que en la comparativa con los 23 algoritmos testeados por (Minella, 2014) nuestro algoritmo obtiene resultados competentes. Además se pone más en valor la importancia de la calibración la cual supone mejorar a 1 algoritmo (en el modo permutation) y a 4 algoritmos (con el modo non-permutation).

Estos resultados van en la línea también de los obtenidos en (Gutiérrez, 2016), donde se concluía que los resultados en la optimización monobjetivo de makespan eran comparables con los mejores algoritmos encontrados en la literatura.

# 6. Conclusiones

## 6.1 Conclusiones finales

Tras el análisis de las experimentaciones realizadas se pueden obtener distintas conclusiones interesantes sobre el funcionamiento del algoritmo ACO tanto en optimizaciones monobjetivo (makespan) como multicriterio (makespan y tardiness).

En primer lugar, en el análisis monobjetivo y sin apenas calibrar el algoritmo, se consiguen mejores soluciones si los problemas de flow shop se resuelven en modo permutation. En este caso, los valores están cerca, o incluso mejoran, respecto los obtenidas durante el Trabajo de Fin de Grado, donde se calibró en gran medida el algoritmo para la resolución de problemas en modo non-permutation.

Sin embargo, también se ha observado que, en algunos problemas de los estudiados, el mejor resultado obtenido es el primero que prueba el programa (ordenando todos los trabajos en función de sus tiempos de proceso) y luego no consigue ningún resultado mejor. En cambio, cuando se resolvía el problema en modo non-permutation, esa primera solución obtenida

siguiendo el criterio de los tiempos de proceso daba un resultado muy mejorable que, a lo largo de las réplicas, se iba optimizando.

Tras ese primer test, se ha profundizado en la calibración del algoritmo a la hora de realizar optimizaciones multicriterio, tanto en modo permutation como en modo non-permutation. Como conclusión inicial, y que sirve para futuras investigaciones, cabe mencionar que el uso del hipervolumen y épsilon parece ser el acertado, ya que los valores obtenidos de los frentes de Pareto para su comparación correspondían con el estudio visual de estos representados en las gráficas.

En este trabajo ha quedado evidente la importancia de una buena calibración de parámetros, como el número de hormigas utilizadas, el valor de q0, los valores de $\alpha$ y $\beta$, y la decisión de aplicar mejor local.

- El número de hormigas es uno de los pocos parámetros en los que la calibración difiere según el modo de resolución. En el caso de non-permutation, al igual que se concluyó en la calibración del algoritmo en monobjetivo en el TFG, el número óptimo de hormigas es 5. En el caso de permutation, los resultados mejoran conforme aumenta el número de hormigas hasta llegar a 20.
- En cuanto al valor de q0, el valor idóneo para los dos modos es aportar al algoritmo más aleatoriedad, dando a q0 el valor de 0.2.
- También en ambos modos se ha podido ver que merece la pena dar más importancia a la feromona y menos a la heurística (dando valores altos a $\alpha$ y valores bajos a $\beta$), lo cual ha podido suponer que haya sido más difícil decidir cuál era la heurística más adecuada.
- Por último, así como en la optimización monobjetivo no se apreciaba que la aplicación de la mejora local fuese conveniente a la hora de resolver el problema en modo permutation (ya que esa búsqueda no reportaba ninguna solución mejor que las conseguidas mediante la aplicación del algoritmo base de la colonia de hormigas), en la optimización multiobjetivo sí que se ha comprobado su efectividad, siendo incluso recomendable aplicar la mejora local a 3 o más soluciones.

Por otro lado, ha habido otros parámetros cuya calibración no ha sido tan evidente, pese a que sí que hemos encontrado un rango de valores que parecían indicar que los resultados de la optimización mejoraban:

- Para ambos modos de resolución se ha concluido que es mejor tener un valor de $\varphi$, evaporación de la feromona, grande, lo cual implica que la información de la feromona, en función de si la hormiga de la réplica ha cogido o no ese camino, se actualiza de forma más brusca.
- La actualización a posteriori, para rebajar la información de lo feromona tras cada iteración de las mejores soluciones, y así, evitar la convergencia del algoritmo en óptimos locales, conviene utilizarla en el modo permutation (aplicándola a 5 hormigas y con un valor de actualización de feromona $\rho$=0.5), mientras que en el modo non-permutation es mejor no realizar dicha actualización.
- En cuanto a las heurísticas solo se han podido desechar algunas de ellas según el modo, pero, al estar usando valores de $\beta$ bajos, la influencia de las heurísticas en los resultados es menor.

Con estas calibraciones, se puede observar una mejora en los resultados, del 10% en el valor de hipervolumen en el modo permutation, y del 80% en el caso del modo non-permutation. En el indicador de épsilon, en modo permutation se ha mejorado de un valor de 1.12 a 1.09, y en el modo de non-permutation de 1.29 a 1.19.

Ante esto, primero es importante valorar la evidencia de que se obtienen mejores resultados cuando se resuelven los algoritmos en modo permutation, y, por otro lado, que la calibración afecta mucho más al modo non-permutation y, por tanto, se puede valorar que tiene más potencial de mejora si se continúa calibrando el algoritmo con mayor detalle.

Para finalizar, en la comparación con otros autores, se concluye que los resultados obtenidos son buenos dentro de la literatura, aunque en ningún momento se ha conseguido una solución que pertenezca a los frentes de Pareto óptimos conocidos en la literatura.

## 6.2 Líneas futuras de trabajo

Una de las primeras y más necesarias futuras líneas de trabajo es el análisis de cómo afecta el tiempo de computación en la mejora de los resultados. Durante la realización de (Gutiérrez, 2016) se concluyó que para resolución en modo non-permutation y monobjetivo, los resultados convergían con 10 segundos de tiempo en problemas de 20 trabajos y 5 máquinas.

Este hecho se repite en la optimización monobjetivo en modo permutation, ya que en la monitorización de los resultados de makespan ha mostrado que no había una mejora continua en la obtención de mejores resultados.

Sin embargo, es importante hacer esa valoración en las optimizaciones multicriterio, tanto comprobando si para estos problemas pequeños, 20 trabajos y 5 máquinas, como para problemas más grandes se mejoran los resultados con mayor tiempo de computación, así como estudiar cómo es la convergencia en función del tiempo simulado.

Otra línea de investigación interesante sería corroborar el buen funcionamiento del algoritmo buscando otros objetivos como la optimización de otros atributos, o, incluso, la optimización de más de dos atributos a la vez.

Por otro lado, tanto el TFG como este trabajo han tenido una clara dirección hacia la automatización y mejora de la generación de más simulaciones y su análisis. Continuando con este objetivo una de las líneas que más enriquecería el programa a largo plazo sería la implantación del cálculo de las métricas en el propio código C++.

De esta manera, el algoritmo podría reconocer las configuraciones más adecuadas para el problema que está optimizando y, sin necesidad de crear baterías de experimentos, él solo podría decidir los parámetros más idóneos. Así, las entradas introducidas manualmente para realizar dicha calibración automática serían únicamente los criterios de grado detalle, tiempo de la calibración, y parámetros a calibrar.

Y, por último, pero a su vez la razón más importante del estudio de las capacidades de este algoritmo, sería buscar una aplicación real sobre la que testear este programa y así poder hacer una evaluación global sobre los beneficios obtenidos del uso de este algoritmo en la programación de secuencias de operaciones.

# Bibliografía


Bruker. (2006). *Scheduling Algorithms.*

Dermikol, E. (1998). Benchmarks for shop scheduling problems. *European Journal of Operational Research*.

Dorigo, M. (1992). *Optimization, Learning and Natural Algorithms.*

Dorigo, M. (2006). *An Introduction to Ant Colony Optimization.* IRIDIA-Technical Report Series.

Giovanni Lizárraga, S. B. (2010). El problema de medir la diversidad en conjuntos no dominados y una propuesta de solución. *Rev. Int. Mét. Num. Cálc. Dis. Ing.*, 217-224.

Grunert da Fonseca, V. F. (2001). Inferential performance assessment of stochastic optimisers and the attainment function. *In Evolutionary Multi-Criterion Optimization, First International Conference,*, 213-225.

Gutiérrez, M. J. (2016). Análisis y evaluación de un algoritmo de colonia de hormigas para la resolución de problemas de programación de la producción multiobjetivo.

Hasija, S. y. (2004). Scheduling in flowshops to minimize total tardiness of jobs. . *International Journal of Production Research*, 2289-2301.

López-Ibáñez, M. P. (2006). Hybrid population-based algorithms for the bi-objective quadratic assignment problem. . *Journal of Mathematical Modelling and Algorithms,* , 111-137.

Minella. (2014). *Optimziación multiobjetivo para la programación de la producción.*

Pinedo. (2012). *Scheduling, Theory, Agorithms and Systems.*

Taillard, E. (1989). *Benchmarks for Basic Scheduling Problems.*

Toncovich. (2009). Resolución de un problema de programación de la producción mediante dos procedimientos metaheurísticos bicriterio.

Zitzler, E. (1999). Evolutionary Algorithms for Multiobjective Optimization: Methods.

Zitzler, E. K. (2008). Quality assessment of pareto set approximations. *Lecture Notes in Computer Science*, 373-404.

Zitzler, E. T. (2003). Performance assessment of multiobjective optimizers: an analysis and review. *IEE, Transactions on Evolutionary Computation*, 117-132.

# Anexos

## Anexo 1: Tipos de problemas de secuenciación de operaciones

Partiendo del escenario general anteriormente planteado hay que definir en primer lugar cómo se da el flujo de operaciones. Inicialmente, suponiendo que las máquinas están en serie, (Bruker, 2006) clasifica estos problemas de la siguiente manera:

- Flow shop problems: los trabajos tienen la misma ruta de procesamiento, es decir, todos los trabajos pasan por todas las máquinas.
    - Con Permutación: en cada máquina el orden de los trabajos es el mismo.

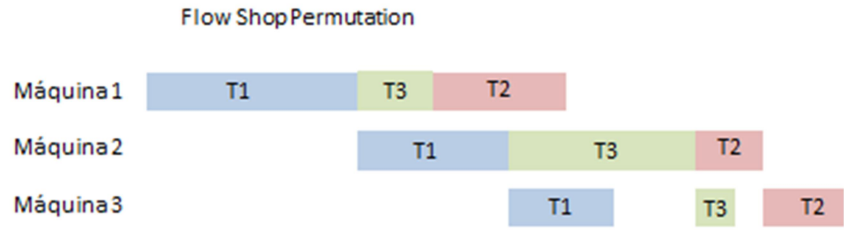



Figura 38: Flow shop con permutación

    - Sin permutación: el orden de los trabajos en cada máquina no tiene porqué ser el mismo.

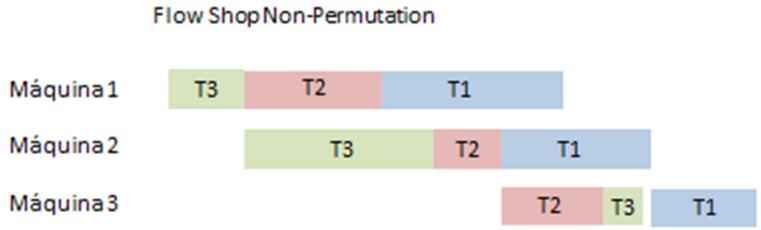



Figura 39: Flow shop sin permutación

- Job shop problems: el orden de las operaciones de cada trabajo está definido, aunque este orden no tiene por qué coincidir de un trabajo a otro.

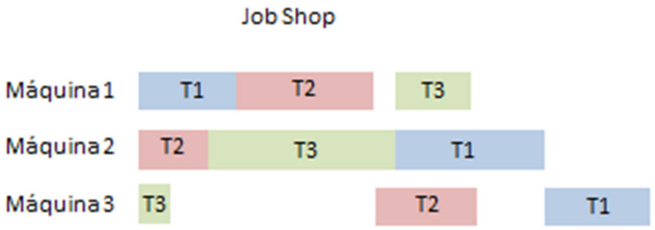



Figura 40: Job shop

- Open shop problems: las operaciones de los trabajos no tienen un orden establecido.

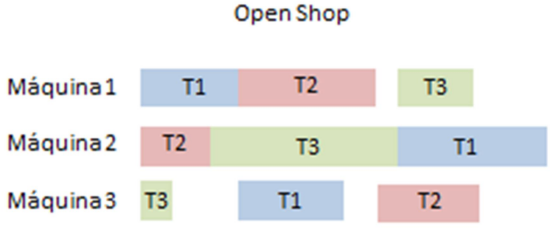



Figura 41: Open shop

- Mixed shop problems: una combinación entre job shop y open shop.

En el caso de tener máquinas paralelas, además de darse estas posibles situaciones iguales que en el caso de máquinas en serie, hay que tener en cuenta también distintos escenarios:

- Máquinas idénticas en paralelo: cada trabajo puede ser procesado en cualquiera de las máquinas y el tiempo del procesado es el mismo en todas ellas.
- Máquinas con diferentes velocidades en paralelo: cada máquina tiene unos tiempos de proceso independientes del trabajo y de la máquina.

Una vez definido el tipo de flujo de operaciones entramos a introducir las restricciones que puede tener un problema, pero para ello cabe definir antes dos tipos de secuenciaciones:

- Secuenciación estática: toda la información del enunciado es conocida al inicio de la simulación.
- Secuenciación dinámica: conforme avanza el proceso productivo se van añadiendo tareas y restricciones con ellas.

## Anexo 2: Atributos posibles de optimizar en problemas de secuenciación

Los atributos que se encuentran en la literatura que son sensibles de ser optimizados en un problema de optimización son los siguientes:

- $C_{max}$, makespan o tiempo de finalización de la última operación programada, es decir, instante de tiempo en el que llega el último trabajo al almacén de producto terminado.
- $F_{total}$, flowtime total o suma del flowtime de cada uno de los trabajos, definido éste como el tiempo desde que el trabajo está disponible en el almacén de materia prima hasta que llega al almacén de producto terminado.
- $F_{MAX}$, flowtime máximo, o valor máximo del flowtime de cada uno de los trabajos.
- $T_{total}$, tardanza total o suma de la tardanza en ser finalizado cada trabajo respecto el límite superior de su ventana de tiempo.
- NumT, número de trabajos retrasados o suma del número de trabajos cuya finalización ha sido posterior al límite superior de su ventana de tiempo.
- $E_{total}$, adelanto total o suma del adelanto en ser finalizado cada trabajo respecto el límite inferior de su ventana de tiempo.
- NumE, número de trabajos adelantados o suma del número de trabajos cuya finalización ha sido anterior al límite inferior de su ventana de tiempo.
- $Desv_{total}$, desviación total o $T_{total}+E_{total}$.
- NumDesv, número de trabajos finalizados fuera de ventana o NumT+NumE.
- $Vol_{MAX}$, la diferencia máxima entre la capacidad de los almacenes y la necesaria medida en volumen.

El algoritmo MACO con el que realizamos la experimentación en este trabajo está diseñado para ser capaz de optimizar cada uno de estos atributos de forma individual y conjunta.

# Anexo 3: Tipos de algoritmos usados para la resolución de problemas de secuenciación

Existen diversos algoritmos utilizados para la resolución de problemas de secuenciación. Dichos procedimientos pueden clasificarse en:

- Métodos exactos: estos procedimientos aseguran llegar a una solución óptima en caso de que esta exista. El problema es que en general no son siempre lo suficientemente robustos, ya que existe una gran variedad de problemas y no hay ninguno que sirva eficientemente para todos y ellos, y, por otro lado, salvo que el problema sea muy pequeño y sencillo, necesitan tiempos de cálculo excesivamente grandes. Varios ejemplos de este tipo de resoluciones son:
  - "Búsqueda exhaustiva": consiste en comprobar todas las posibles soluciones.
  - "Divide y vencerás": se basa en abordar el problema por partes, lo cual en general facilita encontrar la solución al tener problemas más pequeños y luego reensamblar los resultados para dar una solución al problema completo.
  - "Branch and Bound": ampliamente utilizado por los buenos resultados que obtiene, el cual consiste en una búsqueda exhaustiva como la anteriormente definida pero que cuando detecta que un conjunto de posibles soluciones no van a dar una solución mejor al mejor resultado obtenido hasta el momento las veta para no analizarlas y así no necesitar tanto tiempo de simulación.
- Métodos iterativos de aproximación: también llamados métodos heurísticos, se caracterizan por la búsqueda de una solución partiendo de la mejor solución que tienen hasta el momento, estos métodos no aseguran una solución óptima (aunque a veces sí que la consiguen). En general el procedimiento seguido por estos métodos es sencillo, pero sus resultados no son buenos para problemas complejos. Algunos ejemplos de este tipo de resoluciones son:
  - "Búsqueda local": la cual se caracteriza por buscar soluciones vecinas a la que ya tiene, partiendo de una solución, contempla hacerle pequeños cambios, sin con ello consigue una mejora se adopta dicha solución, sino se descarta, pero el riesgo de este procedimiento es estancarse en una solución óptima sólo en un reducido conjunto de soluciones factibles.
  - "Greedy": También llamados algoritmos codiciosos o voraces, estos procedimientos suele combinarse con otros para mejorar su efectividad. Su procedimiento consiste en ir escogiendo, paso a paso, la construcción de la solución más óptima a priori en el momento que se realiza la decisión. Se requiere de poco cálculo computacional, pero, como se ha dicho, utilizando solo este método no se consiguen buenos resultados.
- Algoritmos metaheurísticos: son una evolución de los heurísticos, a los que se les aporta tanto la propiedad de hacer una búsqueda más concreta en un pequeño espacio de soluciones factibles como, por otro lado, la capacidad de no cerrarse a una solución óptima sin ninguna mejor parecida a ella, ya que tienen cierto reseteo y control en la búsqueda de otras soluciones muy distintas a la idónea encontrada. Estos procedimientos son los más conocidos y utilizados por su gran capacidad de enfrentarse a cualquier tipo de problema, por complejo que sea. Algunos ejemplos son:
  - "Genéticos": se basan en la evolución natural de los seres vivos, generando un conjunto de "individuos" que adquieren sus soluciones mediante aleatoriedad o intercambio de información entre ellos y, llegado el momento, se realiza una selección de los mejores y se generan más a partir de ellos.
  - "Búsqueda tabú": es una búsqueda local que, con el fin de no converger rápidamente en una solución óptima local, se prohíbe, en su búsqueda de soluciones, volver a contemplar soluciones óptimas ya contempladas anteriormente.

- “ACO”: Algoritmo basado en la Colonia de Hormigas, que es el algoritmo que utilizamos en este trabajo.

## Anexo 4: Complejidad de los problemas de secuenciación de operaciones

En el trabajo se lleva a cabo la optimización de un problema flow shop con fechas de entrega para cada uno de los trabajos. Sin embargo el algoritmo es capaz de valorar y resolver problemas más cercanos a la realidad y con más complejidad en su planteamiento base. Por ello aquí se detallan tanto aspectos generales del problema como detalles que permiten resolver problemas que no se encuentran en la literatura.

El problema en cuestión, se aparta del caso del flow shop básico, ya que no todos los trabajos tienen operaciones en todas las máquinas. Una posible simplificación consiste en que todos los trabajos pasen por todas las máquinas, aunque el tiempo de operación sea 0. Desde el punto de vista productivo, esto equivale a un equipo multiestación sin flexibilidad alguna. Con esta simplificación, y desde el punto de vista de programación de operaciones, el problema se puede clasificar como un flow shop básico.

En este caso no se va a realizar esta simplificación, de manera que si un trabajo no tiene operación en una máquina, éste no pasará por ella. Por tanto, deben tenerse en cuenta elementos adicionales que llevan a considerar una configuración productiva de mayor complejidad teórica. Por otro lado, la metodología empleada para abordar el problema está basada en un algoritmo metaheurístico multiobjetivo.

Los principales rasgos que caracterizan el problema modelizado y resuelto se han clasificado en dos apartados. En el primero se describen las características más generales del problema, y en el segundo, los elementos que diferencian el problema resuelto de otros tipos.

Características generales:

Como ya se ha comentado, el problema consiste en establecer la secuencia óptima en la que M-máquinas *(l=1…M)* van a procesar la operaciones de N-trabajos *(i=1…N)*, cada uno de ellos formados por un lote de piezas de tamaño variable,

Así, el sistema está compuesto por un almacén de materia prima al que llegan los trabajos en un instante conocido pero no fijo para todos los trabajos. Estos trabajos están formados por un número determinado de piezas llamado lote. Los trabajos serán secuenciados en aquellas máquinas donde tengan que realizarse operaciones, de manera que antes de entrar al equipo esperarán en el almacén de entrada al equipo. Cuando finalice la operación, las piezas del lote se colocarán en el almacén de entrada de la siguiente máquina donde deben sufrir otra operación. Finalizadas todas las operaciones, el trabajo acabará en el almacén de producto terminado.

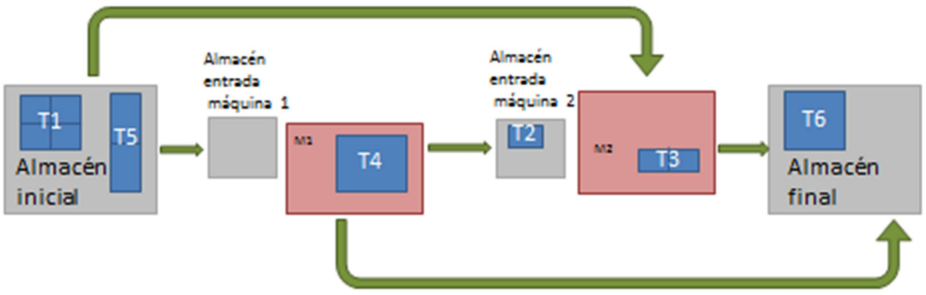



Figura 42: Esquema flujo trabajos

Aspectos diferenciadores del problema:

En este apartado se introducen los elementos más característicos y a la vez más diferenciadores del tipo de problema que se intenta resolve.

- Se considera que la secuencia de operaciones de una máquina a otra no tiene porqué ser la misma. Además si un trabajo no tiene operación en un equipo, aquél no pasará por dicha máquina.
- No todos los trabajos están disponibles en t=0, además de que los tiempos de entrada al sistema pueden ser diferentes. Las piezas que forman un trabajo entran al sistema en el mismo instante de tiempo, es decir, el lote entra al completo.
- No existe prioridad de procesamiento entre los lotes de piezas en cada máquina.
- El almacén de materia prima o de entrada es de capacidad finita.
- Tanto el almacén final de piezas terminadas como los almacenes intermedios de producto en proceso dispuestos previamente a las máquinas se consideran que tienen una capacidad ilimitada (ya que, aunque se cuente con la información de la capacidad de los almacenes, dicho dato no supone una restricción en la generación de las secuencias).
- El tiempo de traslado de cada pieza de una máquina a la siguiente se considera despreciable o en todo caso está incluido en el tiempo de procesamiento de la pieza. Sin embargo sí se tiene en cuenta el tiempo de traslado desde el almacén inicial de materias primas hasta cada una de las máquinas del proceso.
- Las piezas de un mismo lote se procesan de forma consecutiva, y una vez que comienza el procesamiento de un lote, éste no puede interrumpirse hasta que no se hayan terminado de procesar todas las piezas de dicho lote. Además, cada pieza debe procesarse a lo sumo en una única máquina en un instante de tiempo dado.
- Igualmente, cada máquina sólo puede procesar una única pieza. Todas las máquinas están disponibles en todo momento para procesar las piezas cuando sea necesario.
- Por otro lado, cuando la pieza de un lote sale de la máquina en donde está siendo procesada ésta pasa a estar disponible para comenzar otro proceso en la siguiente máquina (sin necesidad de que el resto de piezas de su lote termine en la máquina anterior).
- Los trabajos tienen asociada una ventana de tiempo de entrega en la que deben estar finalizados, intentado no estar antes del límite inferior ni después del límite superior de dicha ventana.
- Se consideran diferentes tipos de preparación de máquina:
  - Setup asociado al cambio de trabajo en una máquina: el tiempo está compuesto por un tiempo fijo, independiente de la secuencia (propio de cada máquina y del tipo de pieza) y un tiempo variable, dependiente de la secuencia en cada máquina. Este tiempo es depende del trabajo que se ha finalizado, del trabajo que se va a iniciar y de la máquina en cuestión. Esta tarea se inicia en cuanto la máquina ha finalizado la tarea anterior, sin tener que esperar a que llegue la primera pieza del siguiente trabajo.
  - También se define este setup cuando se inicia la actividad de la máquina, dependiendo en este caso del equipo y del primer trabajo que procesa.
  - El tiempo de preparación entre piezas de un mismo lote es nulo o despreciable.
  - Setup asociado al desgaste: tras un tiempo en funcionamiento del equipo o tras haber realizado un número de piezas, la máquina sufre una puesta a punto. Esta tarea se ejecuta antes de iniciarse el trabajo.

- Cada pieza tiene definido un volumen, de manera que se podrá evaluar la ocupación de los almacenes o buffers intermedios tanto en unidades como en volumen, intentando no sobrepasar en ningún momento el límite medido en estas 2 unidades.
- El análisis se ha limitado al caso estático y determinístico, de forma que todos los datos son conocidos al inicio de la programación y además no presentan variabilidad.

Así, los datos del problema o variables de entrada son:

- Máquinas: número de máquinas y disposición (orden en el que serán visitadas por los trabajos).
- Trabajos: número, tamaño de lote (o número de piezas que lo forman), instante del tiempo en el que están disponibles para ser procesados (todas las piezas del lote entran a la vez a almacén de materia prima), límites de la ventana de tiempo en la que tienen que ser entregados y volumen de cada pieza.
- Relación máquina-trabajo:
  - Operaciones: tiempo de operación de cada pieza del j-trabajo en *l*-máquina.
  - Tiempo de reconfiguración de la *l*-máquina cuando ésta pasa de procesar el *j*-trabajo al *i*-trabajo.
- Almacenes: capacidad máxima en tiempo y volumen.

## Anexo 5: Modelo matemático del problema

Esta sección describe el problema bajo análisis utilizando un lenguaje matemático y presenta la formulación de programación matemática multicriterio resultante, que puede resolverse utilizado un software de optimización general. Debido a su complejidad, en el modelo sólo tienen en cuenta lotes de piezas de tamaño unitario. Por otro lado, a la hora de formular el problema multiobjetivo se ha optado por introducir el tiempo máximo de finalización o makespan y el flowtime total o tiempo de permanencia de todos los trabajos en el sistema. Parte de este modelo se puede encontrar en (Toncovich, 2009).

Los parámetros del modelo se definen de la siguiente manera:

$p_{jl}$: Tiempo de proceso de una pieza del trabajo j en la máquina l ($p_{jl} \geq 0$), j = 1,…,n, l = 1,…,m.

$r_j$: Tiempo de llegada del trabajo j ($r_j \geq 0$), j = 1,…,n.

$v_j$: Volumen de una pieza del trabajo j ($v_j > 0$), j = 1,…,n.

$s_{ij}^l$: Tiempo de preparación relacionado con el cambio del i-trabajo al j-trabajo en la l-máquina ($s_{ij}^l \geq 0$), i = 0,…,n (i = 0 en el caso de que j sea el primer trabajo de la secuencia), j = 1,…,n, i ≠ j, l = 1,…,m.

$n_l$: Número total de piezas que deben procesarse en la máquina l, l = 1,…,m.

$y_j$: Número total de operaciones que deben realizarse al trabajo j, j = 1,…,n.

V: Capacidad en volumen del almacén.

M: Un número muy grande.

ε: Un número positivo muy pequeño.

Las variables de decisión del problema se describen a continuación con los siguiente subíndices: *i, j, k=1,…,n*; *l = 1,…,m*; *o=1, 2,…,$y_j$*.

$P_{kl}$: Tiempo de proceso de la operación en la posición k de la secuencia en la máquina l.

$S_{kl}$: Tiempo de preparación de la operación asignada a la posición k en la máquina l.

$B_{kl}$: Tiempo de inicio de la operación en la posición k de la secuencia en la máquina l.

$B_j$: Tiempo de inicio de la primera operación del trabajo j.

$C_{kl}$: Tiempo de finalización de la operación en la posición k de la secuencia en la máquina l.

C*j*: Tiempo de terminación de la última operación del trabajo j.

$$X_{jkl}: \begin{cases} 1 & \text{si el trabajo } j \text{ se asigna a la posición } k \text{ de la secuencia en la máquina } l, \\ 0 & \text{en cualquier otro caso.} \end{cases}$$

$$U_{ijkl}: \begin{cases} 1 & \text{si el trabajo } j \text{ se asigna a la posición } k \text{ de la secuencia y está precedido por el trabajo } i \text{ en la máquina } l, \\ 0 & \text{en cualquier otro caso.} \end{cases}$$

$$H_{ji}: \begin{cases} 1 & \text{si el trabajo } i \text{ se ha comenzado a procesar antes de la llegada del trabajo } j, \\ 0 & \text{en cualquier otro caso.} \end{cases}$$

$$A_{jol}: \begin{cases} 1 & \text{si la operación } o \text{ del trabajo } j \text{ debe procesarse en la máquina } l, \\ 0 & \text{en cualquier otro caso.} \end{cases}$$

$$G_{ij}: \begin{cases} 1 & \text{si el trabajo } i \text{ ha llegado antes o al mismo tiempo que el trabajo } j \ (r_i \le r_j), \\ 0 & \text{en cualquier otro caso } (r_i > r_j). \end{cases}$$

Así, el modelo de programación matemática queda como sigue:

$$\text{minimizar} \quad \left\{ f_1 = \max_{j=1}^{n} \{C_j\},\ f_2 = \sum_{j=1}^{n} (C_j - r_j) \right\} \qquad (1)$$

sujeto a:

$$\sum_{j=1}^{n} X_{jkl} = 1 \quad k = 1,\ldots,n_l \text{ y } l = 1,\ldots,m, \qquad (2)$$

$$\sum_{k=1}^{n_l} X_{jkl} = \sum_{o=1}^{y_j} A_{jol} \quad j = 1,\ldots,n \text{ y } l = 1,\ldots,m, \qquad (3)$$

$$\sum_{i=1}^{n} \sum_{j=1}^{n} U_{ijkl} = 1 \quad k = 2,\ldots,n_l \text{ y } l = 1,\ldots,m, \qquad (4)$$

$$X_{jkl} + X_{i(k-1)l} - 1 \le U_{ijkl} \quad i \text{ y } j = 1,\ldots,n, (i \ne j),\ k = 2,\ldots,n_l \text{ y } l = 1,\ldots,m, \qquad (5)$$

$$S_{1l} = \sum_{j=1}^{n} s_{0jl} X_{j1l} \quad l = 1,\ldots,m, \qquad (6)$$

$$S_{kl} = \sum_{i=1}^{n} \sum_{j=1}^{n} s_{ijl} U_{ijkl} \quad k = 2,\ldots,n_l \text{ y } l = 1,\ldots,m, \qquad (7)$$

$$P_{kl} = \sum_{j=1}^{n} X_{jkl} p_{jl} \quad k = 1,\ldots,n_l \text{ y } l = 1,\ldots,m, \qquad (8)$$

$$B_j \ge B_{kl} - M(1 - X_{jkl}) \quad j = 1,\ldots,n,\ k = 1,\ldots,n_l \text{ y } l \ni A_{j1l} = 1, \qquad (9)$$

$$B_j \le B_{kl} + M(1 - X_{jkl}) \quad j = 1,\ldots,n,\ k = 1,\ldots,n_l \text{ y } l \ni A_{j1l} = 1, \qquad (10)$$

$$C_{kl} = B_{kl} + S_{kl} + P_{kl} \quad k = 1,\ldots,n_l \text{ y } l = 1,\ldots,m, \qquad (11)$$

$$C_j \ge C_{kl} - M(1 - X_{jkl}) \quad j = 1,\ldots,n, k = 1,\ldots,n_l \text{ y } l \ni A_{jy_j l} = 1, \qquad (12)$$

$$C_j \le C_{kl} + M(1 - X_{jkl}) \quad j = 1,\ldots,n, k = 1,\ldots,n_l \text{ y } l \ni A_{jy_j l} = 1, \qquad (13)$$

$$B_j \ge r_j \quad j = 1,\ldots,n, \qquad (14)$$

$$B_{kl} \ge C_{(k-1)l} \quad k = 2,\ldots,n_l \text{ y } l = 1,\ldots,m, \qquad (15)$$

$$B_{kl} + M(1 - X_{jkl}) \ge C_{pq} - M(1 - X_{jpq}) \quad j = 1,\ldots,n, o = 2,\ldots,y_j, l \ni A_{jol} = 1, q \ni A_{j(o-1)q} = 1, k = 1,\ldots,n_l \text{ y } r = 1,\ldots,n_q, \qquad (16)$$

$$-r_j + B_i \le M\left(1 - H_{ji}\right) \quad i = 1,\ldots,n \text{ y } j = 1,\ldots,n, \tag{17}$$

$$r_j - B_i \le MH_{ji} - \varepsilon \quad i = 1,\ldots,n \text{ y } j = 1,\ldots,n, \tag{18}$$

$$\sum_{i=1}^{n} v_i G_{ij} - \sum_{i=1}^{n} v_i H_{ji} \le V \quad j = 1,\ldots,n, \tag{19}$$

$$X_{jkl} \in \{0,1\} \quad j = 1,\ldots,n,\ k = 1,\ldots,n \text{ y } l = 1,\ldots,m, \quad U_{ijkl} \in \{0,1\} \quad i = 1,\ldots,n,\ j = 1,\ldots,n,\ k = 1,\ldots,n_i \text{ y } l = 1,\ldots,m, \tag{20}$$

$$H_{ji} \in \{0,1\} \quad i = 1,\ldots,n \text{ y } j = 1,\ldots,n, \tag{21}$$

Ec. 1 determina los criterios del problema: minimización del makespan y del flow time. Ecs. 2 y 3 exigen, por un lado, que en cada máquina sólo haya asignado un trabajo (la operación correspondiente) en cada posición de la secuencia, y, por el otro, que cada operación de un trabajo esté asignada a una única posición de la secuencia de su máquina correspondiente.

Ecs. 4 y 5 aseguran que sólo el trabajo j se coloca después del trabajo i cuando éste ha sido asignado a la posición k–1 en la máquina l. Ecs. 6 y 7 determinan los tiempos de preparación para la primera posición de la secuencia y las posiciones subsecuentes en la máquina l. El tiempo de procesamiento del trabajo situado en la posición k de la secuencia en la máquina l viene dado por ec. 8.

El inicio de la primera operación sobre el trabajo j se calcula mediante ecs. 9 y 10. Ec. 11 define el tiempo de terminación de la operación asignada a la posición k de la secuencia en la máquina l. El tiempo de terminación de la última operación del trabajo j se obtiene a partir de ecs.12 y 13. Ec. 14 restringe el inicio de la primera operación del trabajo j, de manera que éste sea mayor o igual que su tiempo de llegada. Por otro lado, el inicio de la operación asignada a la posición k de la secuencia en la máquina l debe ser mayor o igual que el tiempo de terminación de la operación previa de la secuencia en la misma máquina (ec. 15). Ec. 16 requiere que el comienzo de la operación *o* del trabajo j en la máquina l sea mayor o igual que el tiempo de terminación de la operación previa del mismo trabajo. Ecs. 17 y 18 permiten determinar si el trabajo i ha comenzado a ser procesado antes de la llegada de del trabajo j. La limitación en el valor de la capacidad del almacén de alimentación queda definida con la ec. 19.

Por otro lado, las ecs. 20 y 21 definen las variables binarias del modelo.

## Anexo 6: Uso del dashboard

La programación de esta herramienta ha ido orientada tanto a mejorar el análisis de los resultados obtenidos tras las experimentaciones como para diseñar un modo de uso sencillo e intuitivo.

Por tanto, para dar la información necesaria al documento Excel para que nos represente la información que queremos necesitamos copiar en la misma carpeta que se encuentra el documento 3 ficheros que general nuestro programa de optimzación, llamados “CONFIGURACIONES.txt”, “RESULTADOS.txt” y “Makespans.txt”, y, un cuarto, que contiene la información de los tiempos de proceso de cada trabajo en cada máquina, que se llama “fichero_datos_de_Planificacion.txt”.

Una vez que tenemos esos cuatro ficheros en la misma carpeta que el documento Excel abrimos el archivo y vamos a la pestaña de “Monocriterio” o “Bicriterio” (en función del análisis que queramos realizar).

Tras ello, pulsamos en el símobolo de la EINA que se encuentra arriba a la izquierda Figura 43, y con ello el archivo realiza todas las funciones de extracción de información, análisis y

visualización. Según el número de soluciones y el grado de precisión que se haya definido en el cálculo de épsilon este proceso puede tardar varios minutos.

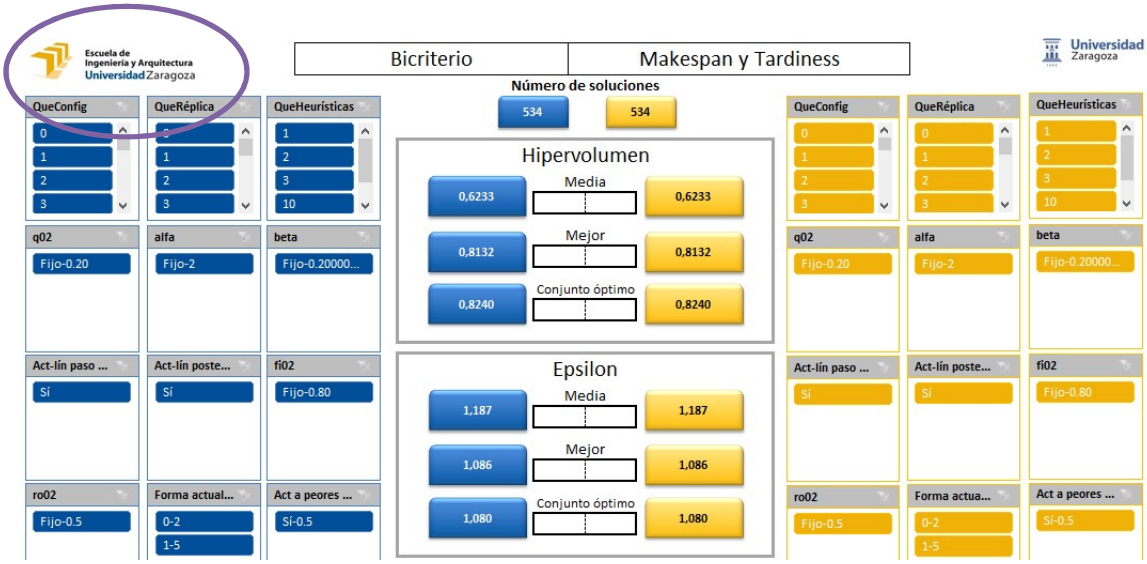



Figura 43: Pulsar en el símbolo de la EINA para hacer funcionar el dashboard.

Una vez terminada la macro basta con pulsar los filtros del dashboard para poder comparar entre los conjuntos.

Para visualizar en la gráfica los puntos de los paretos óptimos de ambos conjuntos y el óptimo ideal hay que pulsar, tras seleccionar los filtros que deseemos, en el círculo azul que se encuentra a lado del gráfico (Figura 44):

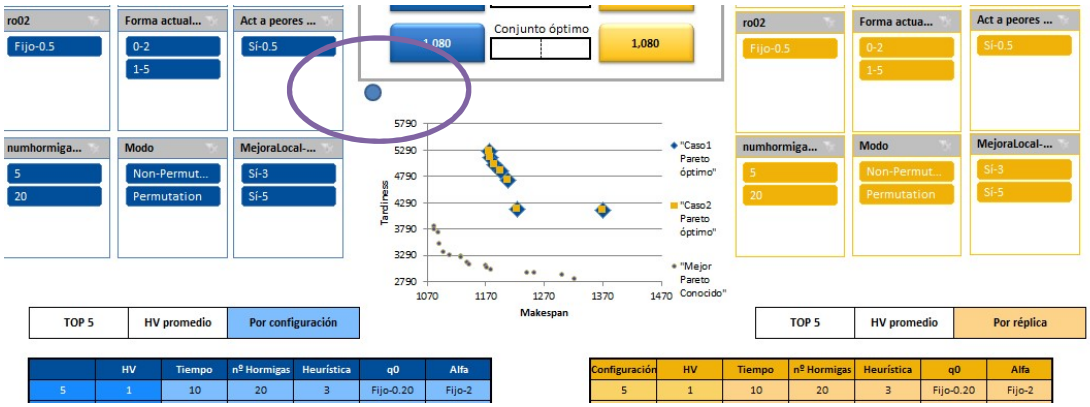



Figura 44: Pulsar botón azul circular cada vez que queramos representar de nuevo los frentes de Pareto.

## Anexo 7: Explicación técnica del dashboard

Con el fin de facilitar la comprensión de la estructura de la programación que tiene el dashboard para posibles futuras mejoras se va a explicar el orden que sigue la macro para realizar todos los pasos que la llevan a visualizar la información que necesitamos para el análisis de las simulaciones.

La macro principal con la que se realiza el análisis bicriterio se encuentra en el módulo 3, y llama a las siguientes funciones:

```
Sub BiCriterio()

Application.DisplayAlerts = False
Application.ScreenUpdating = False
    Application.StatusBar = "Calculando..."
    ExtraerDatosBlocNotas
    Application.StatusBar = "10%... Extrayendo blocs de notas"
    LlevarResultadosAConfiguraciones
    Application.StatusBar = "20%... Calculando Hipervolumen"
    CalculoHiperVol
    Application.StatusBar = "30%... Creando mejor Pareto"
    CreacionDeMejorPareto
    Application.StatusBar = "40%... Calculando Hiperovolumen de mejor pareto"
    CalculoHiperVolMejorPareto
    Application.StatusBar = "50%... Calculando Epsilon"
    CalculoEpsilon
    Application.StatusBar = "60%... Creando tablas dinámicas"
    CrearTablaDinGeneralBiCriterio
    Application.StatusBar = "70%... Creando Paretos óptimos de ambos casos"
    RecreacionDeMejorPareto
    Application.StatusBar = "80%... Creando gráfica de Parets"
    CrearGraficaParetoOptimo
    Application.StatusBar = "80%... Creando Top5"
    TablaDinGraficasMejores
    CeldasDeTopEnPpal
    Application.StatusBar = "Finalizado"

Application.DisplayAlerts = True
Application.ScreenUpdating = True

End Sub
```

Figura 45: Macro principal

Con el primer comando inhabilito la visualización de avisos por parte de Excel (que aparecen en ocasiones cuando se elimina una hoja o una tabla dinámica) y congelo la pantalla para que la visualización de cada uno de los pasos no ralentice la velocidad de éstos.

Mediante el comando “Application.StatusBar” se va reflejando en la barra inferior la información que quiero que aparezca, en este caso se va mostrando el paso que se está realizando conforme avanza la macro.

Con las dos primeras funciones realizo la extracción de información del bloc de notas al Excel, convierto dicha información en una tabla dinámica, y añado campos, en función de otros campos propios de la tabla, para visualizar mejor los valores de los parámetros de las configuraciones.

El cálculo de hipervolumen se hace para cada una de las réplicas de cada una de configuraciones. Para ello, mediante un ‘for’ se va recorriendo la tabla entera, cada vez que se detecta que comienza una réplica nueva se recogen los resultados de la réplica anterior, se pasan a otra hoja Excel, y ahí se calcula el hipervolumen, considerando los valores máximos para calcular el área los máximos encontrados en toda la simulación o los introducidos a mano. Para hacer el cálculo se requiere ordenar el conjunto de soluciones que forman el Pareto en orden de uno de los dos criterios, y uno a uno se va sacando el área respecto los valores de ese punto con los máximos y con los valores del siguiente punto.

```
For i = 0 To NumeroSoluciones
    If i = 0 Then
        HV = HV + (MaxCriterio1 - Cells(6 + i, 6)) * (MaxCriterio2 - Cells(6 + i, 7))
    Else
        HV = HV + (MaxCriterio1 - Cells(6 + i, 6)) * (Cells(5 + i, 7) - Cells(6 + i, 7))
    End If
Next
```

Figura 46: Cálculo de hipervolumen

A continuación se obtiene la mejor frontera de Pareto con el conjunto de todas las soluciones. Para ello se cogen todos los resultados, se ordenan respecto un criterio (criterio 1), se considera el primer resultado que está dentro de la frontera de Pareto, y luego se recorren los resultados de arriba abajo hasta encontrar una solución que tenga un valor del criterio 2 menor (si es que se optimiza buscando una minimización del criterio 2) que el valor del criterio 2 del primer resultado que ya pertenece a la frontera de Pareto, y se continua así con el resto de las soluciones, hasta obtener una frontera de Pareto global.

```
For i = 0 To NumeroDeResultados

    If i = 0 Then
        Cells(5 + m, 8).Value = Cells(5 + i, 5).Value
        Cells(5 + m, 9).Value = Cells(5 + i, 6).Value
        xmin = Cells(5 + i, 6).Value
        m = m + 1
    Else

        If Cells(5 + i, 6).Value < xmin Then
            Cells(5 + m, 8).Value = Cells(5 + i, 5).Value
            Cells(5 + m, 9).Value = Cells(5 + i, 6).Value
            xmin = Cells(5 + i, 6).Value
            m = m + 1
        End If

    End If


Next
```

Figura 47: Creación de frente de Pareto

Tras tener ese Pareto óptimo ya podemos obtener el valor de épsilon para cada una de las fronteras de Pareto de cada iteración. Para ello se van comprobando diferentes valores de épsilon (empezando por 1, y aumentando mediante un factor de aumento que se puede calibrar para que el algoritmo sea más rápido o más preciso) con todas las soluciones de la frontera de Pareto que estamos calculando y comparando el resultado (la división de los puntos de la frontera entre épsilon) con los puntos de la frontera de Pareto óptima.

```
FactorDeAumento = 0.01
Epsilon = 1

For k = 0 To 100

If k = 10 Then
FactorDeAumento = FactorDeAumento * 10
ElseIf k = 20 Then
FactorDeAumento = FactorDeAumento * 5
ElseIf k = 30 Then
FactorDeAumento = FactorDeAumento * 2
End If


Epsilon = Epsilon + FactorDeAumento

For i = 0 To NumeroSoluciones
    Cells(6 + i, 9).Value = Cells(6 + i, 6).Value / (Epsilon)
    Cells(6 + i, 10).Value = Cells(6 + i, 7).Value / (Epsilon)

    For j = 0 To NumeroSolucionesOptimo
    Cells(6 + i, 10).Select
    Cells(6 + j, 3).Select
    Cells(6 + i, 9).Select
    Cells(5 + j, 2).Select
        If j = 0 And Cells(6 + i, 9).Value < Cells(6 + j, 2).Value Then
        k = 100
        ElseIf j = NumeroSolucionesOptimo And Cells(6 + i, 10).Value < Cells(6 + j, 3).Value Then
        k = 100
        End If
        If (Cells(6 + i, 10).Value < Cells(6 + j, 3).Value And Cells(6 + i, 9).Value < Cells(7 + j, 2).Value) Or (Cells(6 + i, 9).Value < Cells(6 + j,
        k = 100
        Else
        End If
    Next
Next

Next

Epsilon = Epsilon - FactorDeAumento
```

Figura 48: Cálculo de épsilon

Tal y como está configurado el código se ha diseñado para que comience la comprobación con mayor precisión, y conforme avance el algoritmo y no se obtenga el valor de épsilon la precisión va bajando a cambio de que se consiga el resultado final más rápido.

Por último se realizan las tablas dinámicas necesarias para mostrar la información en la ventana final (promedios, mejores resultados y top 5), y se crean los segmentadores (filtros) con las conexiones necesarias a todas las tablas dinámicas para que el dashboard sea dinámico.